UNIVERSIDAD DE CUENCA

FACULTAD DE INGENIERÍA

ESCUELA DE INFORMATICA

Título:

“APLICACIÓN WEB PARA LA GESTIÓN DE MICROCRÉDITOS”

Tesis previa a la obtención del título de Ingeniero de Sistemas.

Autores:

Ronald Marcelo Gualán Saavedra

Diego Fabián Montúfar Cevallos

Director:

Ing. Jorge Mauricio Espinoza Mejía

Cuenca- Ecuador

Abril 2012

Agradecimientos

El agradecimiento principal es para mis padres, quienes con su amor, apoyo y sabios consejos han sabido guiarme por el camino del bien y la excelencia infundiéndome grandes valores especialmente la responsabilidad y entereza. Los logros de toda mi vida y éste el más grande de todos los he podido conseguir gracias a ellos y la fe que siempre han tenido en mí.

Un enorme gracias al Ing. Iván Villavicencio, por su confianza y significativos aportes para el desarrollo de esta tesis. De igual manera a Juan Almeida, un gran amigo que compartió con nosotros sus conocimientos y experiencia, que fueron de gran ayuda para el desarrollo de este proyecto.

Un agradecimiento especial al Ing. Mauricio Espinoza, nuestro director de tesis, quién desde el principio nos apoyó y dio ánimo para que pudiéramos completar con éxito esta importante meta.

Ronald

Agradecimientos

A mi familia, que han sido el soporte a lo largo de mi carrera Universitaria y quienes han estado incondicionalmente a mi lado en la espera de conseguir una de las metas más importantes en mi vida. De manera especial a mi padre quien me dio todo su apoyo y a pesar de las dificultades siempre estuvo conmigo.

Agradezco a Dios, en quien confié y puse mi fe, asimismo me dio la fortaleza, sabiduría e inteligencia para lograr conseguir todo lo que tengo y soy.

A Juan Almeida, un gran amigo que nos ayudó a empezar este proyecto y fue nuestra guía durante todo el proceso. Al Ing. Mauricio Espinoza quien también tuvo parte importante en el mismo y finalmente a mis amigos y amigas quienes son lo más valioso que tengo y de una u otra manera han estado cerca dándome su apoyo y aliento.

Diego

Dedicatoria

A mi familia, especialmente a mi madre, que me brindo su cariño y apoyo incondicional toda mi vida especialmente durante mi formación universitaria, sabiendo darme ánimo y alegría en todo momento, sobre todo en los momentos más difíciles y adversos.

Ronald

Dedicatoria

A mi madre, quien a pesar de la distancia nunca dejó de estar pendiente de mí, con sus palabras de aliento me motivó a seguir adelante, con todo su amor y comprensión me hizo saber que no estaba solo. Este trabajo va dedicado a ella como muestra de agradecimiento y como un logro más que ha conseguido su hijo quien está muy orgulloso de ella.

Diego

Abstract

The aim of this thesis is to create a tool to manage the necessary processes to commercialize microcredits. For this, firstly we plan to develop a robust platform that leverages the benefits of using a transactional process management, modular system architecture and the use of recent technologies in the management of persistence and development of Java Web interfaces such as JPA and GWT respectively. With the obtained results we pretend to give support to microcredit Assessors to carry out their work in remote areas of the city and also getting the product to offer credits to customers who do not have facilities for these kinds of loans through traditional means. Microcredits are a means of development and economic growth at these sectors, which are generally poor. The microcredit assessors require tools such as laptops, smartphones or tablets to do their field work. This web application is intended to run on these devices, especially laptops and tablets, making it easier for microcredit Assessors to deal with all the process. This paper discusses the processes flow that involves proper management of a microcredit product, as well as the organizational structure and aspects relating to marketing, sales, and monitoring considered in that flow. Then, we analyze topics related to the technological tools used to develop the application. We present characteristics, advantages and disadvantages, along with statistical analysis to justify the use of each one of them. It also addresses a formal analysis of functional requirements of the application, to continue with the analysis, design and implementation of the system. Finally we consider possible future works and the conclusions drawn at the end of the project.

Resumen

El objetivo de esta tesis es crear una herramienta que permita gestionar los procesos necesarios para la comercialización de microcréditos. Para esto primeramente se busca desarrollar una plataforma robusta que aprovecha los beneficios de utilizar un manejo transaccional de procesos, una arquitectura modular del sistema y el empleo de tecnologías recientes en el manejo de persistencias y desarrollo de interfaces web en Java como son JPA y GWT respectivamente. Con los resultados obtenidos se pretende dar soporte a los Asesores de microcrédito que ejercen su trabajo en zonas alejadas de la ciudad, llegando a ofrecer el producto crediticio a clientes que no disponen de facilidades para obtener estos créditos por medios tradicionales. Los microcréditos representan un medio de desarrollo y crecimiento a nivel económico para estos sectores, que generalmente son de bajos recursos. Los asesores requieren de herramientas como computadores portátiles, teléfonos inteligentes o tablets para realizar su trabajo de campo. El presente sistema tiene como objetivo funcionar sobre dichos dispositivos, especialmente sobre computadores portátiles y tablets, facilitando a los asesores la gestión de microcréditos. En el presente documento se analiza el flujo de procesos que implica un manejo apropiado de un producto de microcrédito, así como también la estructura organizacional y aspectos referentes a la comercialización, venta, y seguimiento que implica dicho flujo. Luego se abordan temas referentes a las herramientas tecnológicas a utilizar para desarrollar la aplicación. Se presentan características, ventajas y desventajas, junto con análisis estadísticos para justificar el uso de cada una de ellas. Además se aborda formalmente un análisis de requerimientos funcionales de la aplicación, para así continuar con la parte del análisis, diseño e implementación de la misma. Finalmente ponemos a consideración posibles trabajos futuros y las conclusiones que se obtuvieron al terminar el proyecto.

Contenido

Índice de Ilustraciones

Índice de Tablas

UNIVERSIDAD DE CUENCA

Fundada en 1867

Yo, Diego Fabián Montúfar Cevallos, reconozco y acepto el derecho de la Universidad de Cuenca, en base al Art. 5 literal c) de su Reglamento de Propiedad Intelectual, de publicar este trabajo por cualquier medio conocido o por conocer, al ser este requisito para la obtención de mi título de Ingeniero de Sistemas. El uso que la Universidad de Cuenca hiciere de este trabajo, no implicará afección alguna de mis derechos morales o patrimoniales como autor.

Diego Montúfar
0105362651

UNIVERSIDAD DE CUENCA
Fundada en 1867

Yo, Diego Fabián Montúfar Cevallos, certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor/a.

Diego Montúfar.
0105362651

FACULTAD DE INGENIERIA
UNIVERSIDAD DE CUENCA
SECRETARIA

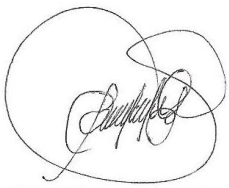
UNIVERSIDAD DE CUENCA
Fundada en 1867

Yo, Ronald Marcelo Gualán Saavedra, reconozco y acepto el derecho de la Universidad de Cuenca, en base al Art. 5 literal c) de su Reglamento de Propiedad Intelectual, de publicar este trabajo por cualquier medio conocido o por conocer, al ser este requisito para la obtención de mi título de Ingeniero de Sistemas. El uso que la Universidad de Cuenca hiciere de este trabajo, no implicará afección alguna de mis derechos morales o patrimoniales como autor.

Ronald Gualán
1400658611

UNIVERSIDAD DE CUENCA
Fundada en 1867

Yo, Ronald Marcelo Gualán Saavedra, certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor/a.



Ronald Gualán
1400658611

PARTE 1
MARCO CONTEXTUAL

Como primera parte de esta tesis se presenta una visión general del tema de estudio, partiendo de la necesidad que cubrirá la realización de este proyecto y definiendo su alcance, considerando aspectos referentes a la arquitectura de Software así como las herramientas de desarrollo.

Capítulo 1
Introducción

El sector microfinanciero latinoamericano está atravesando un período de cambio acelerado y con ello se están impulsando nuevos avances en el sector. El microcrédito es una herramienta que algunas entidades financieras brindan para que las personas puedan desarrollar emprendimientos productivos, comerciales o de servicios de manera asociativa y/o familiar. Está especialmente pensada para quienes no cuentan con garantías patrimoniales o no reúnen las condiciones para acceder a créditos tradicionales. [1]

Este tipo de créditos generalmente se ofrecen a comunidades o grupos de personas en lugares de difícil acceso tanto tecnológico como logístico [2]. Los actores medulares de esta área son los asesores de microcrédito, cuya tarea es la de visitar poblados rurales para ofrecer las oportunidades de microcrédito disponibles, es decir la mayor parte de las tareas de los asesores se realiza como trabajo de campo. Por lo tanto, el problema radica en la dificultad que tienen dichos asesores para desempeñar sus funciones en estos nichos de mercado.

El presente proyecto pretende dar solución a esta problemática a través del desarrollo de una aplicación web diseñada para dispositivos de tipo Tablet y computadores portátiles. Esta aplicación será la encargada de manejar el proceso de planificación, comercialización y seguimiento de los microcréditos. Es decir, se intenta facilitar las labores de las personas que trabajan en el área de microcréditos, mediante una herramienta de software disponible para un dispositivo móvil.

1.1. Justificación

Actualmente en nuestro país, los microcréditos tal vez no son opciones tan conocidas. Esto probablemente se deba a la falta de publicidad e incentivo por parte de las entidades financieras locales. Sin embargo, experiencias internacionales han demostrado que este tipo de créditos de reducida cuantía, pueden proporcionar beneficios importantes tanto para las entidades que las brindan, como para las poblaciones en las que se llevan a cabo. Como uno de los beneficios sociales se destaca el mejoramiento de vida de las personas, debido al incremento a largo plazo en el ingreso y consumo. [3]

Por otra parte, los microcréditos como medios de soporte para comunidades vulnerables, usualmente localizadas en el área rural, requieren de herramientas informáticas de reducido tamaño y portabilidad que faciliten su ejecución. Es así que, los dispositivos móviles como las tablets resulten como muy buenas alternativas, ya

que además de las mencionadas características, poseen excelentes capacidades de almacenamiento procesamiento y variadas opciones de conexión.

En resumen, por medio de la presente tesis se busca solventar la necesidad de un sistema informático que facilite las operaciones vinculadas a la comercialización de microcréditos, y de esta forma poder contribuir e impulsar al desarrollo de esta área.

1.2. Objetivos

Para la presente tesis se propuso como objetivos a alcanzar los mencionados a continuación:

1.2.1. Objetivo General

- Diseñar y desarrollar una plataforma robusta junto con una aplicación web que permitan gestionar los procesos necesarios para la comercialización de microcréditos.

1.2.2. Objetivos Específicos

- Proporcionar una herramienta de software para facilitar la labor tanto en aspectos logísticos como en la comercialización de créditos ofrecidos por los asesores de microcrédito.
- Contribuir al desarrollo de los microcréditos en el medio, de manera que se pueda facilitar el acceso de los clientes a los mismos.
- Fomentar la generación y crecimiento de nuevas microempresas en los sectores beneficiados con este tipo de créditos.
- Crear una plataforma para aplicaciones web con soporte para computadores personales y dispositivos tipo Tablet, que implemente una arquitectura modular y comunicación XML y JSON entre los componentes
- Crear una aplicación web que use un manejo transaccional de operaciones, y que brinde una interfaz gráfica agradable y eficiente.
- Utilizar el framework GXT para el desarrollo de las interfaces Web.
- Utilizar el framework EclipseLink que utiliza tecnología JPA para el manejo de persistencia para la base de datos.

1.3. Alcance

El presente proyecto consiste en el desarrollo de una aplicación web con soporte para computadoras personales y dispositivos tipo Tablet. Posee una arquitectura en capas, que permite la separación de los diferentes componentes que participan en el sistema.

Para impulsar la escalabilidad se tiene un funcionamiento transaccional de las operaciones.

El sistema deberá contener un mecanismo de interoperabilidad, que le permita interactuar con un Core Bancario[1], mediante el intercambio de archivos JSON y XML. Cabe mencionar que no se utilizará un Core Bancario en particular, sino que se empleara un emulador.

Se ha pensado en una interfaz gráfica atractiva, funcional y acorde a los estándares más recientes, para esto se utilizará la plataforma Ext GWT (Sencha). También se empleará EclipseLink, la solución JPA (API de Persistencia de Java) de la Fundación Eclipse, para el manejo de la capa de base de datos.

1.3.1. Arquitectura de software

Se utiliza una arquitectura en capas que permite un manejo independiente y organizado de los diferentes niveles de procesamiento. En la Figura 1.1 se muestra un gráfico que describe la arquitectura de software a ser empleada.

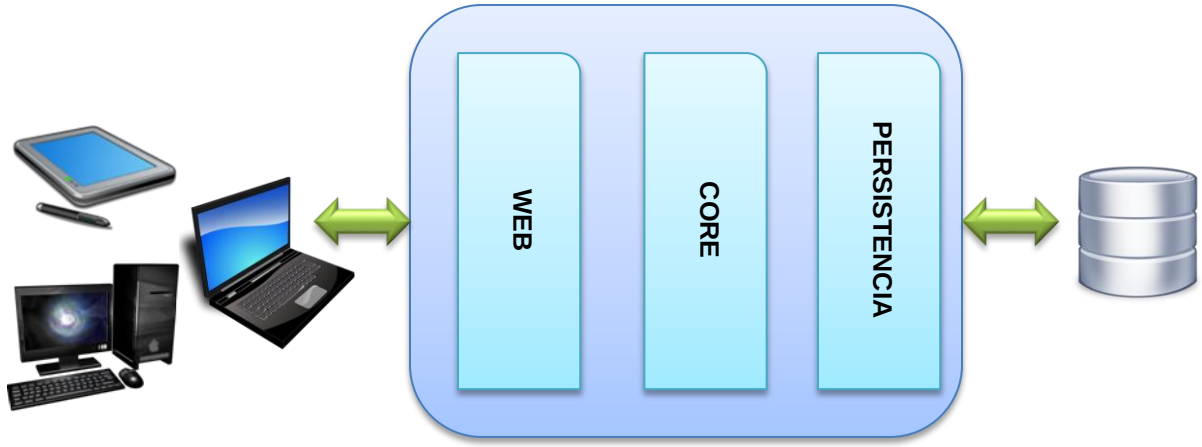


Figura 1.1 Arquitectura de la Aplicación

Se pueden apreciar los tres módulos principales que serán los encargados de manejar todos los aspectos considerados en el proceso de gestión de un sistema de microcréditos y que interactúan entre sí. El proceso empieza con requerimientos enviados desde el cliente, dicho cliente se puede conectar desde una PC o un dispositivo móvil como un Tablet. A continuación se describe brevemente la tarea que realiza cada módulo.

[1] **Core Bancario:** Las soluciones integrales bancarias (core bancario) administran y controlan los procesos y actividades bancarias de las entidades financieras [17].

- **Módulo Web:** En este módulo se incluirá el desarrollo del front-end. Por tanto, es el encargado de recibir las peticiones del usuario, transformarlas en elementos de procesamiento, enviarlos a procesar al módulo Core y devolver los resultados. En este caso el usuario tiene acceso desde un dispositivo tipo Tablet o desde una computadora personal.

- **Módulo Core:** Es el módulo de procesamiento central, encargado de llevar a cabo las tareas principales de procesamiento y de la parte del negocio. Algunas de estas son el levantamiento de la persistencia, manejo de errores, establecimiento de servicios, manejo de la persistencia, procesos básicos y principalmente de los procesos relacionados con la lógica del negocio, además de la interconexión con otros sistemas.

- **Módulo de Persistencia:** Es responsable del manejo de la base de datos mediante el uso de JPA (API de Persistencia Java). Esta tecnología cubre la brecha existente entre los esquemas del modelo de domino orientado a objetos y los sistemas de base de datos relacionales. Entre las ventajas del uso de esta plataforma constan el manejo de objetos en lugar de tablas, portabilidad, lenguaje de consultas, entre otros.

1.4. Auspicio Docente

Director de Tesis: Ing. Mauricio Espinoza

1.5. Recursos

Para el desarrollo de la presente tesis se empleará una cantidad relativamente pequeña de recursos, siendo los más importantes y afortunadamente gratuitos los recursos de software.

1.5.1. Hardware

Entre los equipos de hardware a ser necesarios constan:

- Computadores de trabajo
- Equipos de red
- Dispositivos móviles para las pruebas (principalmente tablets y smartphones).

1.5.2. Software

La aplicación y la plataforma serán desarrolladas en su totalidad con Software Libre, entre las tecnologías utilizadas están:

- GWT (Sencha): Para la parte web, presentación de pantallas e interfaces, controles del cliente, etc.
- EclipseLink con JPA: Para la parte de la persistencia, manejo de la parte de negocio con la base de datos, etc.
- MySQL: Para la administración de la Base de Datos.
- SVN, Google Code y Maven para el trabajo en grupo y control de versiones.
- Entornos de desarrollo como Eclipse, Netbeans y JDeveloper.
- Programas para gestión de base de datos como DbVisualizer
- Emuladores de dispositivos móviles.

1.5.3. Humano

El recurso humano responsable de esta tesis estará integrado por los responsables de la tesis y el representante de Soft Warehouse S.A., quien hará las veces de consultor.

- Ronald Gualán (Responsable)
- Diego Montúfar (Responsable)
- Ing. Iván Villavicencio (Consultor)

1.6. Metodología de Trabajo

El modelo de desarrollo de software será considerado bajo un marco de trabajo que promueve iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto, utilizando el llamado Desarrollo Ágil de Software (APM, Agile Project Management). La metodología para desarrollar el proyecto estará basada en El Proceso Unificado Ágil (AUP, Agile Unified Process) enfoque simplificado del Proceso Unificado de Rational (RUP) de IBM. El ciclo de vida de AUP es serial en lo grande e iterativo en lo pequeño, liberando entregables incrementales en el tiempo [4].

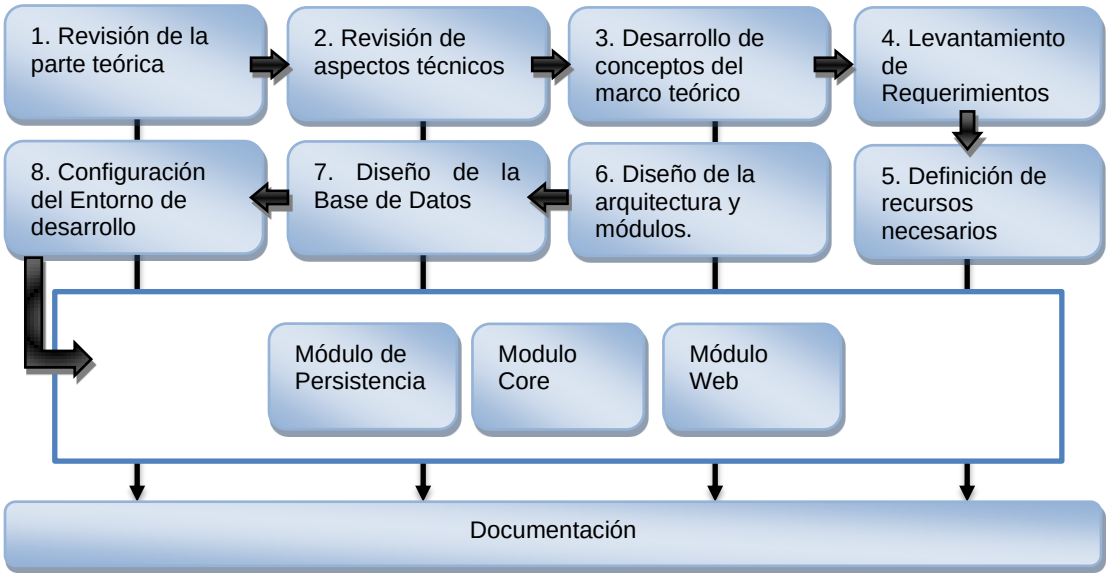


Figura 1.2 Metodología a seguir en el Proyecto

En la Figura 1.2 se muestra el flujo que se seguirá en la metodología del presente proyecto. El mismo comienza con una revisión general sobre los temas a tratar, tanto la parte financiera (estándares, aspectos legales, elementos, actores, procesos, etc.), como la parte técnica (tecnologías, lenguajes de programación, plataformas y herramientas de desarrollo). Así se llevará a cabo la elaboración de la documentación referente a la parte teórica de la tesis, proceso que será revisado y completado en el transcurso del desarrollo de la misma.

Teniendo bases teóricas bien definidas, se llevará a cabo reuniones con nuestro consultor para determinar los requerimientos funcionales de la aplicación. Estas reuniones se deberán realizar con regularidad, de manera que ser pueda definir y supervisar el desarrollo de los requisitos necesarios para diseñar la arquitectura y la aplicación de microcrédito como tal.

La siguiente fase será la del desarrollo de cada uno de los módulos de la aplicación. Este es un proceso iterativo y se lleva cabo en paralelo. Al avanzar en el desarrollo de los módulos de Persistencia, Core y Web, estos se verifican por medio de pruebas de funcionalidad, proceso establecido según el cronograma para determinar el correcto funcionamiento de cada módulo independiente y al final evaluar los mismos con la aplicación completa.

PARTE 2
MARCO TEÓRICO

Con la segunda parte se presenta una revisión general de los conceptos básicos de un programa de microcrédito, luego se hace un análisis de las metodologías de préstamo y se aborda los procesos implicados. El capítulo se centra en explorar el desarrollo de la metodología individual y para ello se consideran aspectos como: La determinación de mercado objetivo y análisis de mercado, la comercialización del Microcrédito, la estructura organizacional de la unidad de microcrédito y finalmente las etapas del ciclo de crédito que deben seguirse para ejecutar adecuadamente los procesos que conlleva esta metodología.

Capítulo 2
Microfinanzas y Aplicaciones financieras

Se estima que en el mundo 500 millones de personas ganan su sustento diario y mantienen a sus familias con el producto de su trabajo independiente. Tejedores, carpinteros, comerciantes, cerrajeros, mecánicos, costureras, jardineros, pintores y muchos otros generan la mayor parte de la fuerza laboral de los países en desarrollo. Un porcentaje muy alto de estos trabajadores son mujeres y una gran mayoría son personas de bajo nivel educativo y muy poco capital. Generalmente no tienen acceso a crédito y en muchos casos ni siquiera tienen facilidades para mantener su dinero en una institución confiable [5].

Aunque los montos que se manejan en los microcréditos son relativamente bajos, permiten que quienes se benefician de ellos puedan adquirir mercadería, herramientas, o equipo para su negocio, generando así ingresos para mantener a su familia.

El desarrollo de instituciones de microfinanzas ha abierto una puerta a estos microempresarios a sistemas modernos y adecuados de crédito y ahorro. Existe conciencia que la mayor limitación para conceder préstamos a este segmento no es el capital sino la falta de capacidad institucional y humana para entregar servicios de microfinanzas. Por esta razón, invertir en el desarrollo de instituciones y áreas especializadas es una manera efectiva de desarrollar esa capacidad y apoyar efectivamente al desarrollo social y económico de nuestros países [5].

2.1. Definición y justificativos técnicos financieros.

A continuación se menciona algunos de los conceptos básicos relacionados con los procesos de gestión y comercialización de microcréditos.

2.1.1. Microfinanzas

El concepto Microfinanzas se refiere a todo un enfoque de desarrollo económico dirigido a beneficiar a hombres y mujeres de bajos ingresos. Siendo las Microfinanzas una forma de apoyar al desarrollo de los microempresarios, debe ser vista como una herramienta de desarrollo social [6].

Como enfoque global, las Microfinanzas incluyen la prestación de los siguientes servicios financieros:

- Pequeños préstamos

- Servicios de ahorro
- Acceso a seguros
- Tarjeta de Crédito
- Servicios de Pago
- Microleasing[2]
- Educación
- Capacitación Técnica y empresarial.

2.1.2. Microcrédito

Es una estrategia contra la pobreza que provee de préstamos pequeños, comúnmente inferiores a USD 100 a personas muy pobres, usualmente mujeres, con el propósito de permitirles ganar ingresos adicionales mediante la inversión en el establecimiento o expansión de micro-empresas tales como las dedicadas a cría de animales, elaboración de alimentos, sastrería, y cientos de otras empresas de similar magnitud [7].

Desde el punto de vista de su destino y características, el Microcrédito puede definirse en base a los parámetros mostrados en la Figura 2.1 y explicado a continuación:

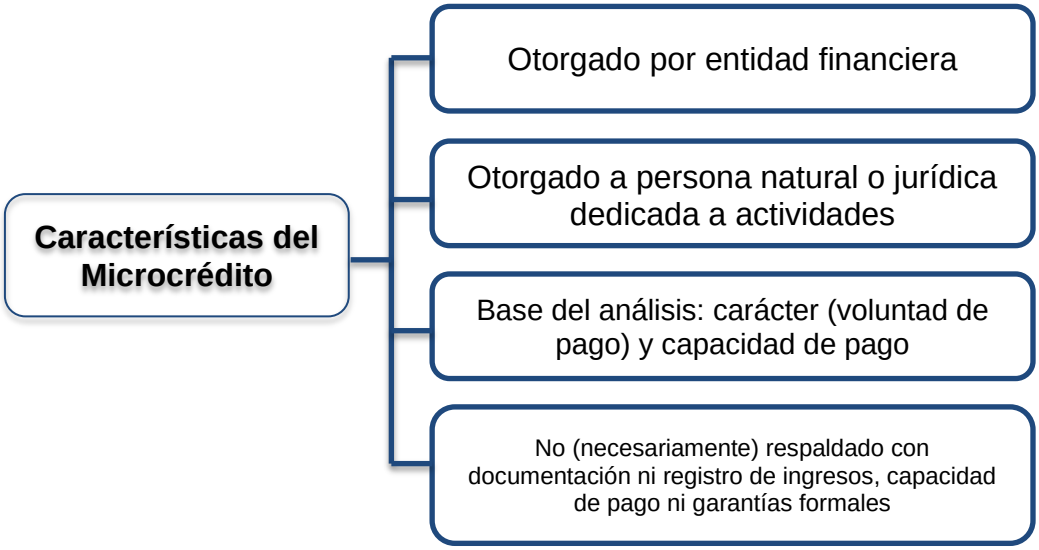


Figura 2.1 Características del Microcrédito

[2] ***El arrendamiento financiero o contrato de leasing*** es un contrato mediante el cual, el arrendador traspasa el derecho a usar un bien a un arrendatario, a cambio del pago de rentas de arrendamiento durante un plazo determinado, al término del cual el arrendatario tiene la opción de comprar el bien arrendado pagando un precio determinado, devolverlo o renovar el contrato [18].

- Es un crédito otorgado por una entidad financiera.
- Es un crédito otorgado a una persona natural o jurídica cuya fuente principal de ingresos proviene de la realización de actividades microempresariales de producción, comercialización de bienes o prestación de servicios.
- No necesariamente cuenta con documentación o registros formales de respaldo sobre los ingresos y la capacidad de pago del solicitante ni con garantías reales registradas.
- Es otorgado sobre la base de un análisis centrado en carácter (voluntad de pago) del solicitante y su capacidad de pago, mediante un análisis del flujo de caja combinado de su negocio y de su hogar.

2.2. Metodologías de Microcrédito

Principalmente existen dos tipos de Metodologías de Microcrédito, como lo indica la Figura 2.2 tenemos: las Metodologías Grupales y las Metodologías Individuales.

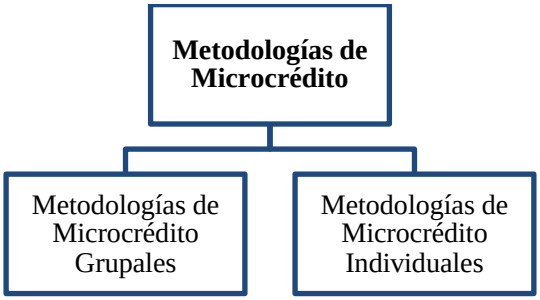


Figura 2.2 Metodologías de Microcrédito

A continuación se describe brevemente la metodología grupal y posteriormente la individual con un poco más de detalle.

2.2.1. Metodología de Microcrédito Grupal.

El Microcrédito Grupal se centra en el principio de la solidaridad grupal. Individuos que debido a su debilidad financiera, falta de información y experiencia son bloqueados del acceso a crédito formal, pueden mediante el trabajo en grupo, constituirse en sujetos de crédito grupales bajo ciertas condiciones.

El modelo de solidaridad grupal es aquel en el que un grupo de individuos garantizan solidariamente los créditos productivos de todos ellos y se apoyan en el control y seguimiento, compartiendo una sola imagen frente a la institución financiera.

En todo el mundo se implementan diferentes metodologías grupales de microcrédito que difieren en menor o mayor grado entre sí. Sin embargo, existen algunas prácticas que se aceptan de manera global como críticas para el éxito del microcrédito grupal:

- **Los grupos más efectivos son de 3 a 15 miembros.** Se utilizan, sin embargo, esquemas que permiten trabajar con grupos mayores de hasta 25 e incluso 50 individuos.

- **Los grupos nominan sus directivos y crean sus propios estatutos.** La autonomía grupal, si bien se apoya en las sugerencias y soporte del oficial de la cooperativa, permite que desde el inicio el grupo asuma el control de su futuro y sea responsable de los resultados.

- **Creación de un "fondo colchón".** Se fomenta en los grupos el ahorro obligatorio, parte del cual servirá de colchón para casos de mora de algún individuo integrante del mismo.

- **Escalonamiento.** Se realizan prestamos que crecen en su monto conforme se renueven los mismos, así se puede empezar con un crédito inicial de montos reducidos desde USD 50 o 60 para primer crédito, Plazos cortos con pagos frecuentes (semanales, generalmente) y un crecimiento paulatino que llegue a un límite máximo determinado y controlado. Para esto se recomiendan niveles de USD 300 a 500.

- **Seguimiento.** El oficial o promotor se constituye también en un asesor permanente para el grupo, tanto en los procesos de control de las reuniones y de manejo del grupo como en la administración financiera.

- **Manejo Interno.** En cada grupo, los directivos (Presidente y Tesorero, generalmente) llevan controles internos en cuanto a ahorros individuales de cada miembro, pago de las cuotas, asistencia y cumplimiento del reglamento interno.

2.2.2. Metodología de Microcrédito Individual.

La Metodología de Crédito Individual es más adecuada para Microempresas de pequeño tamaño. Las diferencias principales consisten en que cada individuo responde por sí mismo. A continuación se presentan los principales Parámetros a considerar previo a la implementación de una unidad de microcrédito individual.

Una Unidad de Microcrédito debe tener claramente definidos sus objetivos y el entorno en el que se desenvuelve, por lo que se deben tener en cuenta los siguientes factores: *el mercado potencial, el mercado objetivo, la economía nacional y regional en que se desenvuelven y las regulaciones que puedan limitar o apoyar el desempeño de esta Unidad*. La Metodología de Crédito es el conjunto de normas y procedimientos que la institución adopta, para otorgar el crédito a un costo razonable y con un nivel de riesgo aceptable.

La aplicación correcta de esta metodología implica la necesidad de información que permita determinar la situación económica financiera del sujeto de crédito (solicitante) para estructurar adecuadamente las condiciones del crédito, realizar el seguimiento y actuar adecuadamente en el caso de no pago.

Es necesario aquí definir las regulaciones que condicionarán el funcionamiento de la Unidad de Microcrédito. Se recomienda tener en cuenta los siguientes aspectos:

- Determinación de: un mercado objetivo, productos y servicios, estrategias de mercadeo, tasas y costos, entre otros.
- Estructura organizacional de la Unidad, parámetros de conformación de la misma; perfiles, competencias, y capitación requeridos para el personal.
- Determinación de mecanismos de compensación variable.

2.2.3. Características del Microcrédito Individual

Las diferentes Metodologías de Microcrédito Individual pueden ajustarse según el mercado en el que se aplique. A continuación se detallan las características básicas que deben estar presentes en dichas metodologías para garantizar el éxito:

- ***Determinación de mercado objetivo.*** Para evitar cartera con riesgos diferentes a los determinados, cuantificados y aceptados por la institución.

- ***Importancia de acceso al crédito para clientes.*** Los socios deberán ser personas o empresas con poco o ningún acceso a crédito formal y que presenten la necesidad del mismo.

- ***Concordancia entre necesidades de clientes y características de productos.*** Los productos a ofrecer serán uniformes, por lo tanto, deberá identificarse aquellos microempresarios cuyas necesidades son adecuadamente cubiertas con los productos ofertados.

- ***Plazos cortos, préstamos sucesivos y escalonados.*** El conocimiento del cliente y el desarrollo de una cultura de pago en el mismo será necesario para poder incrementar paulatinamente montos y plazos de crédito.

- ***Destino adecuado del crédito.*** El inicio de toda relación crediticia deberá ser para el financiamiento de capital de trabajo, ya sea éste inventario o cuentas por cobrar.

- ***Oferta de asistencia técnica.*** El oficial de crédito deberá apoyar al socio a manera de asesor, asistiéndolo permanentemente para una adecuada administración de su negocio, manejo de sus recursos y flujo de caja.

2.2.4. Definición y descripción del mercado objetivo (clientes).

Para una correcta definición del Mercado Objetivo se hace necesario definir claramente las características que identifiquen al tipo de microempresario que se está dispuesto a financiar, teniendo en cuenta los siguientes aspectos:

- Nivel económico o nivel de pobreza al que se quiere llegar.
- Experiencia mínima requerida en su negocio y estabilidad mínima en la ubicación actual de su establecimiento.
- Ubicación geográfica y cobertura. Se debe tener en cuenta que penetrar en zonas geográficas que no se pueda atender adecuadamente, incrementa los costos del seguimiento y reduce la calidad de la supervisión. Hay programas que se desarrollan dirigidos a ciertos grupos en exclusividad por si existe limitación o preferencia por sexo, raza, origen étnico, religión, etc.
- Tipos de actividad apetecidas y tipos de actividad excluidas.

2.2.5. Solicitud de crédito

Mediante este documento, se formalizará la necesidad de un crédito. La solicitud debe indicar como mínimo las condiciones del crédito (monto, tasa de interés, plazo, formas de pago, etc.).

La información declarada en la Solicitud de Crédito deberá estar respaldada por toda la documentación que presente el solicitante y sus garantes con el fin de validar la información.

En esta etapa es importante explicar al solicitante todas las condiciones del tipo de crédito que está solicitando, ya que este documento constituye la base para continuar

el proceso de análisis y posterior recomendación al nivel administrativo de aprobación correspondiente.

2.3. Comercialización del Microcrédito

La comercialización de microcrédito hace referencia al proceso de planificar y ejecutar los planes de socialización y venta de los productos de microcrédito que una entidad ofrece. Es el área principal en la que se desenvuelven los asesores.

2.3.1. Desarrollo de productos y servicios.

En el momento de desarrollar un producto, se debe recordar siempre que "*no vendemos productos ni servicios sino soluciones a necesidades*". [6]. Los socios necesitan los beneficios que les brindará el producto, no el producto en sí ni sus características.

2.3.2. Mercadeo

El Mercadeo es el proceso de planeación y ejecución de la concepción, de las políticas de precios, de las estrategias de promoción y de los mecanismos de distribución de ideas, bienes y servicios con el fin de crear intercambios que satisfagan metas individuales y organizacionales. [6].

Existe 7 aspectos importantes a considerar durante el proceso de mercadeo, estos son referidos como las 7 P's del Mercadeo:

- **Producto.** Este tema ha sido tratado en los puntos anteriores y no es más que la solución que se ofrecerá al cliente. Define las características de los microcréditos a comercializar (montos y plazos permitidos, tasas, formas de pago, requisitos, etc.)

- **Precio.** Su fijación dependerá de la estrategia de Mercadeo. Deberá incluir todos los costos relacionados con la concesión del crédito (tasa, comisiones, trámites, costos relacionados).

- **Promoción.** Es el uso de variados mecanismos de comunicación entre la entidad financiera y el mercado, es decir, sus socios actuales y potenciales.

- **Plaza.** Está relacionada con los canales de distribución y los puntos de venta a través de los cuales se ofertarán los productos de Microcrédito.

- **Personas.** Se refiere al contacto personal con el cliente. Se debe definir los siguientes aspectos respecto al perfil del personal que atienda al socio microempresario:
  - Formación
  - Trato con la gente
  - Entrenamiento y capacitación
  - Forma de vestir, entre otros.

- **Presencia.** Trata sobre la imagen institucional que proyecta la cooperativa, a través de los edificios, muebles, colores, papelería, etc. Debe proyectarse la imagen de una institución seria, formal, sólida y confiable, pero nunca de un lugar de lujo donde un socio de escasos recursos no sea tratado adecuadamente.

- **Proceso.** El proceso es todo lo que sucede durante la prestación del servicio, desde el primer contacto con el socio (o socio potencial) hasta el desembolso y el posterior seguimiento.

2.3.3. Ventas.

La venta de productos de Microcrédito se debe hacer de manera directa, con oficiales de campo cuya función es salir a promocionar puerta a puerta. La labor de los ***Oficiales de Crédito*** se inicia con *la identificación del mercado objetivo, la zonificación del área a cubrir y la delimitación de rutas para su trabajo diario.*

Es necesario que las zonas sean respetadas estrictamente y que se tracen rutas para las visitas y recorridos diarios para evitar que se traslapen zonas entre dos o más oficiales.

2.4. Estructura organizacional de la unidad de microcrédito.

En una Unidad de Microcrédito, al estar conformada dentro de una entidad financiera o en una cooperativa, que atienden a una gran gama de socios diversos, debe existir diferenciación en cuanto a: procedimientos, productos, metodologías de prospección, venta, aprobación, desembolso y seguimiento de los socios.

La Unidad deberá, por tanto, funcionar con autonomía ya que es difícil encajar dentro de la organización de una cooperativa financiera tradicional los procedimientos y controles requeridos para desarrollar un programa exitoso de Microcrédito.

2.4.1. Conformación de la Unidad.

La Unidad de Microcrédito estará conformada por equipos de hasta 8 personas. Cada uno de estos equipos tendrá bajo su la responsabilidad un área determinada. Los equipos estarán liderados por un ***Coordinador*** que reportará al ***Gerente de Crédito*** o al ***Gerente General***, en casos de cooperativas más pequeñas.

2.4.2. Perfiles y funciones.

Coordinador de Microcrédito.

El Coordinador es el funcionario responsable del adecuado cumplimiento de todo el equipo. Supervisa a sus subordinados y responde ante el Gerente General o el Gerente de Crédito por los resultados.

Sus funciones deberán ser las siguientes:

- Coordinar la aplicación correcta de todos los procedimientos controles, y del cumplimiento individual y grupal de su unidad.
- Trazar rutas y delimitar las zonas asignadas a cada uno de sus oficiales, asegurando el adecuado cumplimiento y respeto a la zonificación.
- Apoyar a los oficiales en labores de mercadeo complejas y sobre todo en la cobranza de socios que presenten mayor riesgo para la cooperativa.
- Manejar los contactos grupales, con asociaciones, instituciones y autoridades de su zona.

Asesor de Microcrédito.

El Oficial o Asesor, como se denomina en algunas instituciones, es la posición medular del programa, ya que se constituye en el representante de la institución ante los clientes. Realiza todo el proceso: desde el mercadeo, la venta, la evaluación, la recomendación, hasta la recuperación total del crédito, y luego, repite el ciclo con el represtamo.

Las funciones que contempla este cargo son:

- Administrar el proceso completo de crédito, desde el mercadeo hasta el cobro total de los recursos prestados.
- Promocionar la colocación de recursos en los sectores y socios que desarrollen actividades rentables.
- Recopilar la información relacionada con el cliente en la cooperativa.
- Emitir su opinión sobre la interpretación de la información obtenida.
- Visitar a los clientes para confirmar la veracidad de las informaciones suministradas.
- Mantener información actualizada de los clientes.

- Mantener un sistema de monitoreo efectivo para detectar posibles problemas con el socio.
- Informar a su superior inmediato sobre situaciones de los clientes que puedan perjudicar la recuperación de los recursos prestados.
- Atender a los clientes que presenten algún tipo de problema para cumplir con el acuerdo de pago establecido.

Asistentes de Negocios y Administrativo.

La labor del asistente de Negocios y Asistente Administrativo es proporcionar soporte al resto del equipo. Es el responsables de manejar las áreas de apoyo para liberar a los Oficiales de funciones burocráticas o rutinarias.

Debe cumplir las siguientes funciones:

- Atender a los socios que se acerquen directamente a las oficinas de la cooperativa, proporcionado toda la información correspondiente respecto a productos, requisitos y concertando las citas de inspección para que los Oficiales los atiendan.
- Atender reclamos, consultas y dudas que se presenten en el día a día.
- Manejar todo el proceso operativo, lo cual implica principalmente: el manejo de archivos, la coordinación para el procesamiento y desembolso de los créditos, y la generación de los reportes correspondientes, para el seguimiento por parte del Coordinador y los Oficiales.

2.5. Tecnología Crediticia Aplicada.

Todo proceso de crédito sigue un ciclo que se inicia en la identificación del mercado objetivo y termina con la recuperación final de los valores prestados. En el caso del Microcrédito, existe una variante de gran importancia y es que el ciclo se cierra con el represtamo [5]. Este ciclo se ilustra en la Figura 2.3.Para el Oficial de Crédito, el ciclo tiene las siguientes etapas:

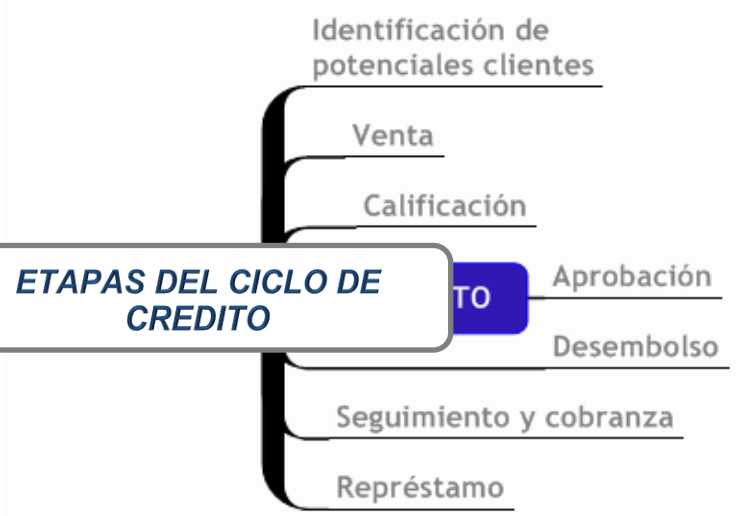


Figura 2.3 Etapas del ciclo de crédito

A continuación se describen cada una de las etapas que cumple un programa de microcrédito.

2.5.1. Identificación de potenciales clientes.

El proceso de identificación de clientes permite al oficial dirigir su actividad de venta hacia los principales centros de agrupación o concentración de clientes potenciales. Este proceso se realiza en cinco etapas:

- ***Análisis macro.*** Se lo realiza en base a mapas de la ciudad y de la zona asignada, buscando concentraciones de zonas comerciales, industriales y de servicios.
- ***Reconocimiento de la zona.*** El reconocimiento es un recorrido rápido que se realiza cubriendo la totalidad del sector asignado, para identificar los sectores de mayor potencial y anotar los principales clientes potenciales que se identifique en cada sector.
- ***Trazado de las rutas.*** El trazado de la ruta se realiza, dando mayor énfasis y prioridad a los sectores con mayor potencial. Se lo hace en conjunto con el Coordinador de la Unidad. Aquí se definen: rutas, frecuencias, cobertura, etc.
- ***Visita promocional.*** Los primeros días se hará un rápido recorrido de las rutas, distribuyendo folletería promocional, tarjetas de presentación e información general de la cooperativa y su servicio de microcrédito.
- ***Recorrido habitual de rutas.*** Una vez terminadas las visitas promocionales, se inicia proceso de la visita y venta puerta a puerta.

2.5.2. Venta.

El proceso de identificación de clientes permite al oficial dirigir su actividad de venta hacia los principales centros de agrupación o concentración de clientes potenciales.

Este proceso es de venta consultiva ya que el oficial debe:

- Generar confianza por parte del cliente.
- Identificar sus necesidades.
- Presentar los beneficios que los productos de microcrédito proporcionarán, así como la manera en que estos podrán satisfacer las necesidades puntuales de su cliente.

2.5.3. Calificación de clientes.

En esta etapa se determinan las condiciones económicas y financieras del solicitante y su unidad familiar. Los resultados que se obtienen en la inspección y evaluación de los solicitantes sirven para determinar su capacidad de pago, evaluando también aspectos cualitativos que permitirán hacer un buen análisis y minimizar el riesgo:

- **Recolección de datos para la evaluación.**

Se parte como punto inicial del formulario de la Solicitud de crédito. Mediante este documento se formaliza la necesidad de un crédito, previo análisis y decisión del tipo de producto de crédito que solicita. La solicitud debe indicar como mínimo las condiciones del crédito (monto, tasa de interés, plazo, formas de pago, etc.).

- **Validación de la información.**

Antes de realizar la interpretación de la información obtenida, se debe analizar su razonabilidad para determinar si es lógica y confiable. Este proceso se realiza en la visita de inspección y también durante todo el trámite en las oficinas de la cooperativa.

Algunos de los documentos exigidos y sometidos a evaluación dentro de este campo son: Documento de Identidad, carta de servicios básicos. A diferencia de los préstamos normales, los requisitos para la concesión de microcréditos no son rígidos.

Uno de los objetivos más importantes a cumplir en esta etapa es la evaluación del Carácter del Socio ya que una prueba de carácter es su honestidad al entregar información completa y veraz. Por esta razón, en esta etapa “si se detecta cualquier falsedad, el crédito deberá ser negado”. Aquí el oficial debe ser muy perceptivo e incluso desconfiado.

2.5.4. Aprobación de la solicitud.

En la etapa de aprobación de la solicitud se cubren una serie de ítems que incluyen la evaluación de la solicitud, una recomendación del crédito y la respectiva aprobación.

- **Evaluación de la solicitud.**

En base a la información recolectada y validada, el asesor de crédito debe realizar la evaluación de la solicitud de crédito. En la evaluación de una solicitud se debe valorar la información proporcionada por el solicitante, interpretando con objetividad la consistencia de la solicitud de crédito, de la entrevista personal y del perfil crediticio presentado. Esto a efectos de establecer la voluntad de pago y la solvencia económica del cliente, para así poder tomar la decisión adecuada para su aprobación o recomendación.

Se evalúa por tanto el *carácter* de la persona, analizando para ello la integridad, el deseo de pagar o las características morales o éticas de los clientes. Además, la *estabilidad familiar* es un aspecto cualitativo muy importante que se debe considerar en el análisis de un solicitante. Los puntos que se deben observar respecto a ésta, son:

- Número de años en el lugar de residencia.
- Número de años en el negocio.
- Número de cargas familiares
- Estado civil (soltero, casado, divorciado, unión libre)
- Referencias de personas que lo conocen, clientes y proveedores.
- Referencia de otras fuentes de ingresos alternativas para poder pagar la cuota en el caso de que se presenten problemas en el negocio
- Referencias personales principalmente de los vecinos y proveedores

Otro criterio considerado es la capacidad de pago, que impone requisitos tales como:

- Buenas referencias comerciales, y de proveedores.
- Buena apariencia del negocio (limpio y ordenado).
- Volumen de ventas de acuerdo al tamaño del negocio y ubicación.
- Destino del crédito a actividades que aseguren el cumplimiento de los pagos.

Luego de la evaluación de la solicitud normalmente se suele hacer una recomendación de la misma.

- **Recomendación del crédito.**

Una vez realizado el análisis de la operación y si el oficial de crédito está satisfecho con el crédito planteado, deberá presentar su recomendación para la aprobación a los

niveles adecuados, en función del monto, plazo y riesgo de la operación, y de ser el caso aprobar directamente la solicitud.

La recomendación de aprobación se realiza en esquemas en donde se maneja otros niveles de aprobación. Pero, para entornos regulares el asesor es el encargado del análisis y aprobación directa.

La recomendación realizada por el asesor de microcrédito deberá incluir, como mínimo, los siguientes elementos:

- ***Comentario de la unidad económica y familiar.*** Se dará una explicación sobre los aspectos generales del negocio y del grupo familiar, con la finalidad de establecer la estabilidad del negocio, del domicilio, familiar (en cuanto a su integración y salud).
- ***Calificación sobre la moral de pago.*** De acuerdo a lo observado en las visitas, se emitirá un juicio de apreciación sobre el nivel de responsabilidad del solicitante con los compromisos que posea tanto en el negocio como en el hogar. Este aspecto es fundamental y, pese a ser subjetivo, deberá sustentarse en la mayor cantidad de argumentos posibles.
- ***Propuesta.*** Se deberá mencionar claramente si se recomienda aprobar o negar el crédito. En ambos casos, ya sea una propuesta de aprobación o negación, se deberá mencionar los argumentos que respaldan la recomendación.

2.5.5. Desembolso de la operación.

Una vez aprobado el crédito se procede al desembolso del dinero. Es la instancia donde concluye el proceso de análisis crediticio y se efectiviza la solicitud del cliente. Además de la entrega de liquidez, en el desembolso se lleva a cabo la implementación de las condiciones y obligaciones adquiridas por el cliente.

2.5.6. Seguimiento y cobranza.

El objetivo que la entidad financiera persigue de recuperar los recursos colocados, solamente se logrará si los fondos que recibió el cliente se invierten en el destino para el cual fueron solicitados, es decir, en una actividad productiva. El destino de los recursos deberá ser para las actividades relacionadas con el giro principal del negocio.

La forma de pago de los créditos estará de acuerdo con la capacidad de pago del solicitante, el plazo asignado y las condiciones pactadas. La entidad financiera establece la posibilidad de dar créditos bajo diferentes periodos de pago, en función de los ciclos de la actividad económica del solicitante. Estos pueden ser:

- Semanal (cada 7 días)
- Quincenal (cada 15 días)
- Mensual (fecha fija o cada 30 días)

Se aplicará el sistema de pago en cuotas fijas e iguales, las cuales están compuestas principalmente de capital e intereses. Los períodos de gracia para pago de capital, se concederán para aquellos clientes que presenten una justificación aceptable relacionada con su actividad, hasta por un plazo máximo de 3 meses. Durante el período de gracia se pagarán únicamente intereses.

- **Seguimiento**

Es imprescindible la supervisión y monitoreo por parte de los asesores de microfinanzas, coordinadores y gerentes para lograr que el cliente cumpla con los compromisos contraídos.

El seguimiento es la herramienta que permite minimizar el riesgo de las operaciones. Aunque un crédito esté bien colocado, esto no evita las contingencias que puedan ocurrir en el transcurso del tiempo, por lo tanto el seguimiento permite tomar medidas preventivas ante cualquier posible riesgo.

La frecuencia con que se lleve a cabo el seguimiento está en función al riesgo. Se realizarán como mínimo 2 visitas de seguimiento durante el plazo del crédito. La primera visita como máximo a los 15 días después del desembolso, para verificar el destino de los recursos. La segunda visita, a la mitad del plazo otorgado o cada seis meses en los créditos con plazos mayores a 1 año.

- **Recuperación**

Independientemente a lo dispuesto por la Superintendencia de Bancos con respecto a la contabilización de la cartera vencida, se considerarán morosos los clientes que tienen un día de vencido en el pago de sus obligaciones, por esta razón el reporte de morosidad, para su control, será emitido diariamente.

Los Asesores de Microfinanzas son responsables de las gestiones de cobro. En caso de morosidad, luego de haber agotado todas las gestiones del caso, se asignará al Oficial de Cobranzas quien primero realizará la cobranza extrajudicial y luego inicia las acciones judiciales correspondientes.

2.5.7. Represtamo.

El socio podrá optar por un incremento en el monto con relación al crédito anterior, siempre que sus necesidades de financiamiento, capacidad de pago, comportamiento de pago del crédito anterior y perspectivas del negocio lo justifiquen. Como política

general, el nivel máximo para un crédito subsiguiente será el equivalente al 1.5 veces del monto anterior.

La ampliación de crédito procederá, siempre y cuando el cliente haya tenido un cumplimiento oportuno en sus pagos.

2.6. Resumen.

En esta sección se presentó una revisión general de los conceptos básicos de un programa de microcrédito junto con una descripción de las dos principales metodologías: la individual y la grupal. Se abordó las diferentes etapas del ciclo del préstamo, las mismas que van desde la determinación de mercado objetivo, la comercialización, pasando por la venta, calificación, aprobación y desembolso, hasta llegar al seguimiento, cobranza y finalmente al représtamo, figurando este último como característica distintiva de este tipo de préstamos. Una vez abordado los conceptos teóricos asociados con los microcréditos se tratarán aspectos de carácter tecnológico que son el tema del siguiente capítulo. Estos conocimientos permitirán desarrollar una herramienta que facilite el desarrollo de una aplicación que maneje los procesos más importantes de la gestión de microcréditos.

PARTE 3
CONTEXTO TECNOLÓGICO

En esta tercera parte se presenta un estudio general de las herramientas tecnológicas a utilizar en el proyecto para desarrollar la plataforma en la aplicación de gestión de microcréditos. Primeramente se hace una revisión sobre el desarrollo de aplicaciones para dispositivos móviles en el medio, considerando algunos análisis estadísticos, a continuación se resumen las características, ventajas y desventajas de las principales tecnologías utilizadas en este proyecto, tales como JPA para la parte del manejo de base de datos, XML y JSON para la comunicación entre el front-end y el back-end, junto con GWT la tecnología a ser empleada para el desarrollo del primero.

Capítulo 3
Desarrollo de aplicaciones para dispositivos móviles

Cuando se plantea por primera vez la creación de una aplicación para smartphones y tablets, normalmente lo primero en lo que se piensa es en la variedad de plataformas existentes: Apple iOS, Google Android, Palm, Symbian, BlackBerry, Windows Phone, etc. Luego la pregunta que surge es ¿cómo realizar un desarrollo que abarque todas o el mayor porcentaje de estas plataformas? Si damos por supuesto que nuestra aplicación debe correr como mínimo en Android e iOS para cubrir una importante cuota de mercado, ¿vamos a dedicar tiempo a desarrollar una misma aplicación en distintos sistemas operativos con el coste de tiempo y formación de un equipo multidisciplinar? Una solución que se plantean muchas empresas ante esta situación, es la realización de una aplicación web para dispositivos móviles, de esta forma con un único desarrollo se puede conseguir el objetivo de llegar a múltiples plataformas y dispositivos.

En este capítulo se tratan los pros y los contras de utilizar aplicaciones nativas y web para dispositivos móviles, considerando que la plataforma a desarrollar servirá para manejar la gestión de un sistema de microcréditos. A continuación se presenta una visión general considerando los puntos clave que se tomaron en cuenta para la misma.

3.1. Aplicaciones móviles nativas vs. Aplicaciones web

El gráfico mostrado a continuación (ver Figura 3.1), compara el promedio de minutos por día que los consumidores gastan en aplicaciones móviles nativas frente a la web. Para aplicaciones móviles, *Flurry,* una compañía analista de tendencias en plataformas móviles, compara a iOS, Android, BlackBerry, Windows Phone y J2ME. Y para la web, las cifras incluyen la web abierta, Facebook y la web móvil.

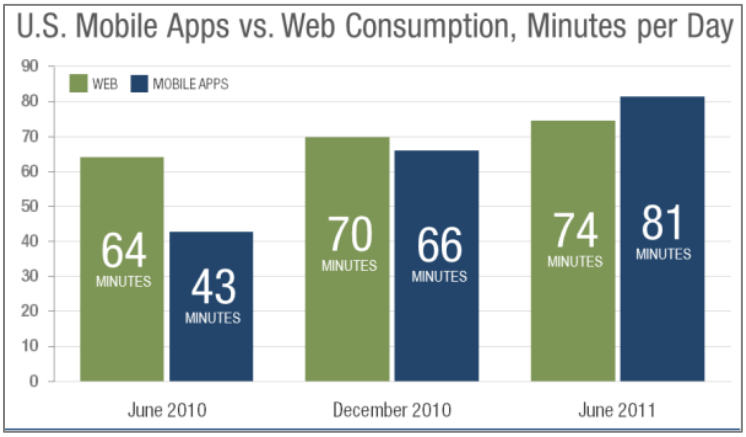


Figura 3.1 Consumo de tiempo de los usuarios en Aplicaciones Móviles y Navegación en la Web [8].

Flurry constató que el usuario promedio pasa un 9% más con aplicaciones móviles que con Internet. Este no era el caso hace sólo 12 meses. El año pasado, el usuario promedio gastaba un poco menos de 43 minutos al día utilizando aplicaciones móviles en comparación con un promedio de 64 en la Internet. Creciendo a 91% en el último año, los usuarios ahora pasan más de 81 minutos en las aplicaciones móviles por día. Este crecimiento ha provenido principalmente de más sesiones por usuario al día en lugar de un gran crecimiento en el promedio de duración de la sesión. El tiempo dedicado a Internet ha crecido a un ritmo mucho más lento, un 16% en el último año, con los usuarios utilizando 74 minutos en el Internet al día.

Las estadísticas indican *que los usuarios prefieren aplicaciones nativas a aplicaciones web para móviles*, y de hecho el tiempo de uso medio de unas y otras es ligeramente superior en el caso de las nativas. Esta situación, sumada a la ventaja inigualable que tiene una aplicación nativa en cuanto al aprovechamiento de hardware y gráficos, hacen que la adopción de aplicaciones nativas solamente vaya a la alza. De hecho, según *Flurry*, el 47% del tiempo promedio en que se tiene abierta una aplicación nativa es para jugar, dejando el 32% para redes sociales, mientras que las noticias, entretenimiento y otras actividades ocupan menos del 10% cada una (Figura 3.2).

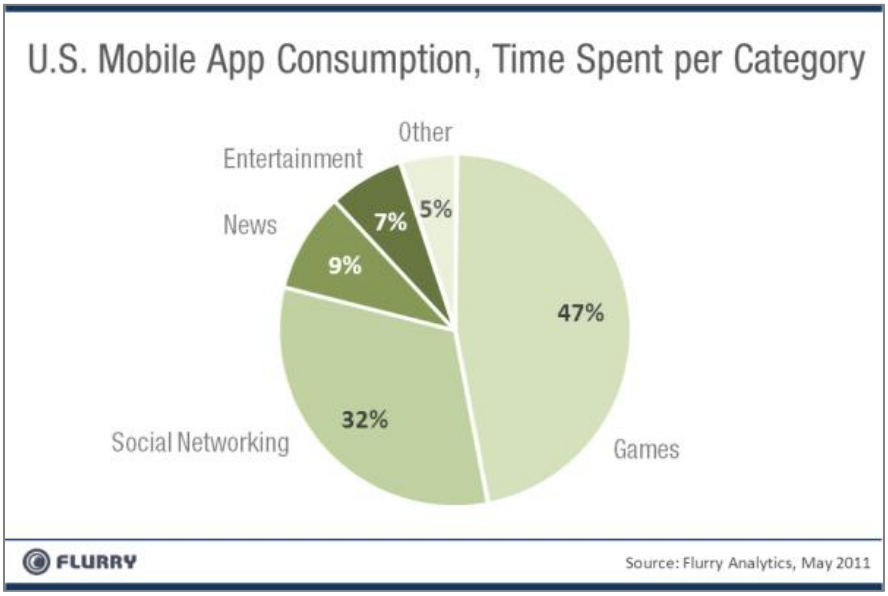


Figura 3.2 Categorías en las que ocupan más el tiempo los consumidores [8]

Esto seguramente se deba a la dificultad para alcanzar el mismo nivel de usabilidad, apariencia e integración con el dispositivo (*acceso al hardware del dispositivo*). Por ejemplo, desde HTML no se puede acceder a algunos elementos particulares como la cámara, GPS, etc. No olvidemos también que las aplicaciones nativas ganan en velocidad de ejecución aunque las mejoras constantes de optimización en el motor de JavaScript V8 han conseguido que la ejecución del código JavaScript de las webs se pongan casi a la misma altura.

Desde hace algún tiempo existe la posibilidad de realizar aplicaciones híbridas: aplicaciones web basadas en HTML5 + JavaScript que son empaquetadas de manera que de un único código fuente de aplicación web tenemos la posibilidad de obtener una aplicación instalable en múltiples plataformas móviles. Existen varias herramientas que lo permiten. Una de las más conocidas y además gratuita es PhoneGap[3]. Con PhoneGap se cuenta además con acceso a gran parte de los elementos del dispositivo (acelerómetro, cámara, GPS, vibración, etc.). En la Figura 3.3 se muestra una ilustración simple donde el código base, escrito en lenguajes de programación web y HTML5 se empaqueta por medio de PhoneGap convirtiendo a dicha aplicación web a una aplicación móvil nativa.

[3] PhoneGap es una plataforma HTML5 que permite crear aplicaciones nativas con tecnologías web y obtener acceso a APIs y tiendas de aplicaciones [9]

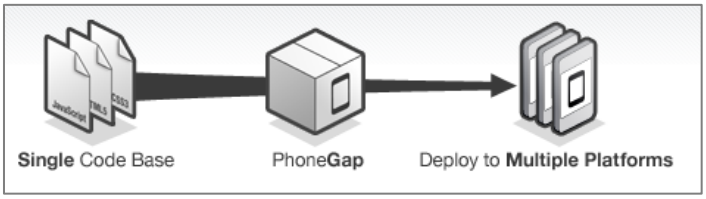


Figura 3.3 Ilustración del esquema de PhoneGap [9]

Sin embargo, cuando hablamos de aplicaciones de tipo empresarial que requieren un análisis más profundo y poseen un nivel de complejidad elevado, las estadísticas mencionadas solo nos dan datos relativos a tendencias de mercado antes que cuestiones de rendimiento, capacidad de procesamiento y almacenamiento. Existen varios aspectos que se deben considerar antes de pensar en desarrollar una aplicación orientada a dispositivos móviles y a continuación hacemos una revisión breve de los mismos.

3.2. Consideraciones para el Desarrollo de Aplicaciones Móviles

El desarrollo de aplicaciones móviles conlleva una variedad de consideraciones de acuerdo al propósito y escenario para el que van a ser utilizadas tales como:

- Tipos de Dispositivos.
- Tipos de Aplicaciones.
- Sistema Operativo.
- Capacidades del dispositivo.
- Navegadores.
- Limitaciones en la conectividad.

A continuación se analiza brevemente a que hace referencia cada uno de estos ítems.

3.2.1. Tipos de Dispositivos

Es necesario primeramente escoger el dispositivo o al grupo de dispositivos a los que irá orientada nuestra aplicación. Se puede clasificar a los dispositivos móviles de acuerdo a la variedad que hoy en día disponemos en el mercado. Algunos de ellos difieren bastante en su funcionamiento, prestaciones y presentación. Los dispositivos móviles con los que contamos en la actualidad se muestran en la Figura 3.4, entre ellos están: *Notebooks, Teléfonos Celulares, Rugged Devices*[4]*, UMPC (Ultra Móvil PC), PDA's y los SmartPhones.*

[4]***Rugged Devices*** (equipos robustos) son equipos diseñados específicamente para funcionar de forma fiable en entornos de uso y condiciones difíciles, tales como fuertes vibraciones, temperaturas extremas y las condiciones de humedad o polvo [18].

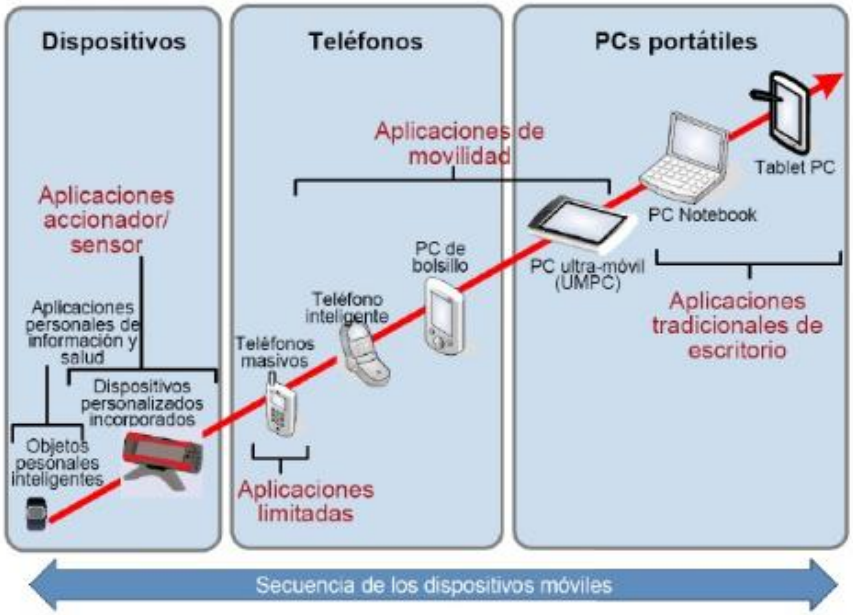


Figura 3.4 Secuencia de los dispositivos móviles

3.2.2. Tipos de Aplicaciones.

Se pueden crear distintas aplicaciones para nuestros móviles, como por ejemplo:

- ***Aplicaciones de sistema***, estas aplicaciones estarán relacionadas al funcionamiento de nuestro terminal, como podrían ser compresores de archivos, seguridad del terminal, registro de llamadas, gestión de mensajes, etc.
- ***Aplicaciones ofimáticas***, son aquellas que permiten trabajar con documentos de texto, hojas de cálculo, ficheros PDF, etc.
- ***Organización:*** este tipo de aplicaciones irán destinadas a la organización de nuestros datos, contactos, notas, etc. Como podrían ser el calendario, gestor de contactos, etc.
- ***Aplicaciones web:*** Para poder utilizar este tipo de aplicaciones hará falta una conexión de Internet en nuestro móvil. Un ejemplo de este tipo de aplicaciones serían: el correo electrónico, Google Maps, navegadores web, etc. Dentro de la categoría anterior podríamos incluir también las aplicaciones relacionadas con las redes sociales, como: Facebook, Twitter, Google+, etc.
- ***Aplicaciones de accesibilidad:*** este tipo de aplicaciones facilitará el uso del terminal para personas con algún tipo de discapacidad. Ejemplos de este tipo de aplicaciones sería: reconocimiento de voz, reconocimiento de caracteres, lectura de texto, etc.
- Existen otros tipos de aplicaciones como las multimedia, los juegos, etc.

3.2.3. Sistema Operativo.

Actualmente el mercado ofrece muy buenas alternativas en este sentido, dos de ellas como se verá más adelante, llevan una ventaja considerable frente a las demás.

- Symbian OS
- Windows Mobile (Windows CE)
- iPhone OS
- Palm OS
- Android
- BlackBerry OS

3.2.4. Capacidades del Dispositivo

Algunas características como: pantalla (tamaño, resolución), memoria, microprocesador, control (t*ouchscreen, directionalpad, teclado primario, etc.), c*ámara, expansión de memoria, batería, navegación, captura (l*ectura de barras, RFID*[5]).

3.2.5. Navegadores y Emuladores:

En la Tabla 3.1 se listan algunos de los navegadores más utilizados en el mercado así también como algunos emuladores que se utilizan como herramienta en el desarrollo de aplicaciones web para dispositivos móviles como smartphones y tablets.

Navegadores disponibles en el mercado	Emuladores de navegadores
• Opera Mobile • Firefox • Internet Explorer Mobile • Android Browser • Skyfire • Dolphin • Netfront • Safari	• WinWap Smartphone Emulator • OpenWave Browser • Nokia Browser Simulator • Microsoft Pocket PC Emulators • Online Mobile Simulator • Online WAP Browser • Offline Emulator • Opera Mini

Tabla 3.1 Navegadores y Emuladores para dispositivos móviles disponibles en el mercado

3.2.6. Limitaciones de Conectividad.

[5] **RFID** (siglas de Radio Frequency IDentification, en español identificación por radiofrecuencia) es un sistema de almacenamiento y recuperación de datos remoto que usa dispositivos denominados etiquetas, tarjetas, transponedores o tags RFID [18].

Un dispositivo móvil puede conectarse ya sea a Internet, alguna red interna empresarial o a otros dispositivos por medio de: *USB, Bluetooth, WI-FI, Infrarrojo, Serial, etc.* Y bajo distintos protocolos de comunicación. Actualmente la conectividad de un dispositivo representa gran parte de su desempeño, ya que un dispositivo que no ofrece buenas características de conexión prácticamente puede pasar desapercibido debido al avance tecnológico y abaratamiento de costos del mercado.

3.3. Las nuevas tendencias en el desarrollo móvil

Todos los años desde el 2008 *Web Directions* lleva a cabo un estudio para evaluar el panorama actual del desarrollo móvil. El objetivo es comprobar la *evolución de las principales plataformas móviles y detectar tendencias en relación al desarrollo de webs y aplicaciones móviles*. El año pasado, el estudio se realizó durante el mes de Febrero entre una muestra de más de 1.300 desarrolladores y diseñadores [10].

A continuación se presentan las principales conclusiones que se extraen de dicho estudio:

- Durante el 2010 se produjo un boom en el interés que los desarrolladores mostraron por el **HTML5**, así como por el desarrollo de aplicaciones nativas para plataformas móviles (como iOS o Android) mediante tecnologías web.
- **Android desbanca al iPhone** como plataforma móvil preferida por los desarrolladores.
- Queda demostrado que existe una correlación clara entre los dispositivos móviles y las plataformas que los desarrolladores usan, y aquellas para las que desarrollan.
- Se confirma que los desarrolladores cada vez más testean sus webs y aplicaciones móviles en una gran variedad de dispositivos y de navegadores móviles.
- **iPhone, iPad y Android (móviles y tablets)** son los dispositivos que centran la mayor parte de la atención de los desarrolladores, abriéndose un destacable hueco con respecto al resto de plataformas y dispositivos móviles.
- A pesar de que los desarrolladores expresan su confianza en la creación de aplicaciones usando HTML5, muestran sus reservas en materia de usabilidad.

Todas estas tendencias quedan resumidas de un modo mucho más visual en la Figura 3.5

State of (mobile) Web Development 2011

Mobile browser of choice:

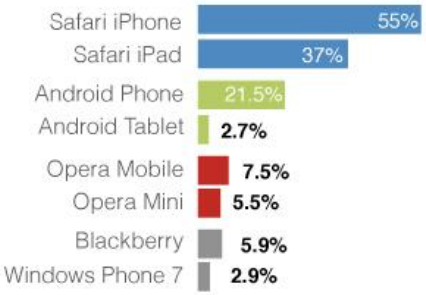
Mobile Safari
23% (29%)

Android
11.8% (3.8%)

Opera Mini
2.3% (2.6%)

Opera Mobile
1.8% (1.4%)

Mobile test browsers:



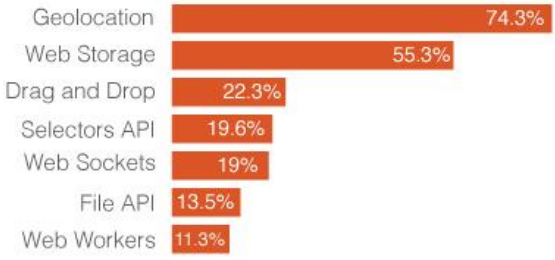
JavaScript Frameworks:

JQuery Mobile 12.5% | JQTouch 6.88% | Sencha Touch 3.6% | Zepto 2%

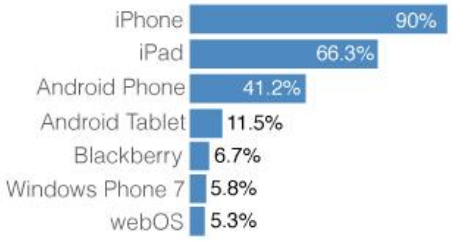
Development Tools:

PhoneGap 47.6% | Appcelerator 26.5% | Adobe AIR 7.8%

HTML5 API Use:



Target Platforms:



162% Increase in developers using HTML5 APIs

Figura 3.5 Web móvil y los resultados en el desarrollo de aplicaciones móviles una encuesta de más de 1.300 profesionales de la web con experiencia, llevada a cabo por WebDirections en marzo de 2011 [10].

A continuación se presenta un análisis breve donde se exponen las principales razones por las que este proyecto hace uso de una aplicación web orientada a dispositivos móviles de tipo Tablet para el desarrollo de la herramienta en la gestión de microcréditos.

3.4. ¿Por qué Tablets?

Un Tablet es un dispositivo compacto que ofrece características similares a las de un ordenador portátil. Los tablets poseen pantallas táctiles y están diseñados para navegar por internet y manejar varias aplicaciones que funcionan de acuerdo a la plataforma instalada en su sistema operativo. Las Tablet PC pueden ser una buena opción para personas que necesitan de portabilidad y que les gusta tener un dispositivo rápido y versátil.

Como podemos ver en el cuadro de la encuesta realizada por *Web Directions (ver* Figura 3.5*)* existe una alta tendencia por los consumidores a utilizar dispositivos *iPhone, iPad y Android*. Y en lo referente a desarrollo para estas aplicaciones vemos que sobresalen tecnologías como la de PhoneGap, JQuery Mobile y Sencha touch.

Este proyecto está enfocado al desarrollo de una aplicación para *comercializar Microcréditos* en zonas rurales, donde el acceso a la tecnología aun representa un problema en nuestro país. Por estas razones se ha considerado buscar la mejor alternativa para desarrollar una aplicación orientada a dispositivos móviles sin perder de vista las tendencias y que sirva de herramienta para ofrecer un microcrédito sin mayores dificultades. Entre otras cosas se pensó que como dispositivo es mejor un Tablet por las siguientes razones:

- Es un dispositivo ligero.
- El entorno táctil hace que en ciertos contextos el trabajo sea más fácil que con el uso de un teclado y un ratón.
- Actualmente poseen una capacidad de almacenamiento razonablemente buena, aunque la de procesamiento aun sea limitada.
- Con su capacidad de conectividad es posible acceder a cualquier recurso vía internet sin mayores dificultades, más que solo las de cobertura de las empresas proveedoras.

3.5. ¿Por qué una Aplicación Web?

Según las tendencias del mercado, las aplicaciones móviles nativas son la opción más escogida por los desarrolladores, sin embargo para fines enfocados a entornos empresariales que necesitan aplicaciones más robustas y grandes, esto no representa la opción más versátil. Para el presente proyecto se necesita manejar una plataforma robusta que requiere una capacidad de procesamiento media-alta, y entre las razones más importantes por las que se pretende manejar una aplicación web pura, para ello se considera lo siguiente:

- No importa para que plataforma se desarrolle, una aplicación web solo depende del navegador donde se ejecute y la conectividad que tenga el dispositivo (*conexión a internet*).
- No es necesario instalar servidores embebidos, como por ejemplo de base de datos o web servers.
- Toda la lógica del desarrollo depende únicamente de la aplicación, mas no del dispositivo. La aplicación se adapta a las características de presentación y del navegador que utiliza este.

Una vez analizado el tipo de aplicación a ser utilizado para este proyecto y el dispositivo principal de enfoque, se aborda una breve revisión de las herramientas tecnológicas para el desarrollo tanto de la plataforma como de la parte de interfaz del cliente que serán utilizadas en el diseño e implementación de la aplicación para la gestión de microcréditos.

3.6. JPA (API de Persistencia Java - Java Persistence API)

JPA es la API de persistencia desarrollada para la plataforma Java EE. Es un framework del lenguaje de programación Java que maneja datos relacionales en aplicaciones usando la Plataforma Java en sus ediciones Standard (Java SE) y Enterprise (Java EE) [11].

JPA cubre tres áreas: La API en sí misma, El lenguaje de consulta JPQL (Java Persistence Query Language) y Metadatos objeto/relacional. El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos y permitir usar objetos regulares (conocidos como POJOs) [12].

Uno de los mayores componentes de JPA, es el mapeo objeto-relación, comúnmente denominado ORM (Object-Relational Mapping), que es la técnica utilizada para eliminar la brecha existente entre el modelo orientado a objetos y el modelo relacional. Sin embargo, JPA también ofrece soluciones para los desafíos de arquitectura e integración de la persistencia en aplicaciones empresariales escalables.

3.6.1. Características de JPA

- **Persistencia con objetos POJO**

Quizás el más importante aspecto de JPA, es que los objetos son POJO (Plain Old Java Objects). Lo que significa que no hay nada especial con los objetos que se pueden hacer persistentes. De hecho, casi cualquier objeto con un constructor por defecto, puede hacerse persistente con solo cambiar una simple línea de código. ORM es enteramente dirigido por metadatos. Los objetos que son persistentes son tan pesados como la información que es definida o mapeada con ellos.

- **API no intrusiva**

La API de persistencia existe como una capa independiente de los objetos persistentes. La API de persistencia es llamada por la lógica de negocio de la aplicación y se pasa los objetos persistentes y las instrucciones para operar sobre ellos. Así que, aunque la aplicación debe ser consciente de la API de persistencia, los objetos persistentes en sí, no necesitan serlo. Esto, debido a que la API no interfiere en el código de las clases de objetos persistentes. De allí, que se llame a este tipo de persistencia como no intrusiva.

- **Consulta de Objetos**

JPA ofrece la posibilidad de consultar a través de entidades y sus relaciones sin tener que utilizar claves externas concretas o columnas de base de datos. Las consultas se pueden expresar en lenguaje de consulta JPQL (Java Persistence Query Language), un lenguaje de consulta que no está ligado al esquema de base de datos. Las consultas utilizan un esquema de abstracción que se basa en el modelo de entidad en lugar de las columnas en las que se almacena la entidad. Se utilizan las entidades java y sus atributos, así que el conocimiento de la información de base de datos no es necesario. Las consultas eventualmente se traducen en SQL y se ejecuta en la base de datos. Una consulta puede ser definida de manera estática en los metadatos o creados de forma dinámica en tiempo de ejecución. También es posible crear consultas SQL, si se requiere hacer consultas especiales que no se pueden manejar desde el framework de persistencia.

- **Configuración sencilla**

Hay un gran número de características de persistencia que JPA ofrece, y todas las funciones son configurables a través del uso de anotaciones, XML, o una combinación de los dos. Las anotaciones ofrecen facilidad de uso sin precedentes en la historia de metadatos Java. Son convenientes para escribir y leer fácilmente. La configuración también se puede hacer en archivos XML para quienes gustan de XML o quiere externalizar los metadatos desde el código.

De mayor importancia que el lenguaje de metadatos es el hecho de que JPA hace un uso intensivo de las configuraciones por defecto. Esto significa que no importa el método que se elija, la cantidad de metadatos que serán necesarios es la mínima. En algunos casos, si los parámetros por defecto bastan, no será necesario metadata en lo absoluto.

Algunas de las implementaciones de JPA 2.0 son DataNucleus, EclipseLink, JBoss Hibernate, ObjectDB, OpenJPA. Las más conocidas de estas implementaciones son EclipseLink y Jboss Hibernate.

- **EclipseLink**

EclipseLink es el proyecto de persistencia de código abierto de la Fundación Eclipse. Este software proporciona un marco extensible que permite a los desarrolladores Java interactuar con los servicios de datos diferentes, incluyendo bases de datos, servicios web, objeto de mapeo XML (OXM), y Sistemas de Información Empresarial (SIE) [12].

EclipseLink apoya una serie de normas de persistencia, incluyendo:

- Java Persistence API (JPA)
- La arquitectura Java para vinculación XML (JAXB)
- Arquitectura de Conector Java (JCA)
- Objetos de Servicio de Datos (SDO).

En cuanto a la implementación JPA de EclipseLink, se puede decir que EclipseLink es una plataforma avanzada de persistencia de objetos y transformación de objetos, que provee herramientas de desarrollo y capacidades en tiempo de ejecución que reduce los esfuerzos de desarrollo y mantenimiento y aumentan la funcionalidad de las aplicaciones empresariales.

3.6.2. Arquitecturas de Aplicación de EclipseLink

Se puede utilizar EclipseLink en una variedad de arquitecturas de aplicaciones, incluyendo la arquitectura Web, SessionBean, y las arquitecturas de dos niveles, con o sin Java EE, para acceder a una variedad de tipos de datos, de fuentes relacionales y no relacionales [12].

Una aplicación web Java es una de las arquitecturas EclipseLink más comunes. Esta arquitectura se caracteriza por un entorno de servidor en el que se aloja la lógica de negocio, las entidades persistentes, y las librerías de EclipseLink, todos existentes en una máquina virtual Java (JVM). El ejemplo más común de esta arquitectura es una aplicación de tres niveles en los que el navegador cliente accede a la aplicación a través de servlets, jsp (JavaServer Pages) y HTML. La capa de presentación se comunica con EclipseLink a través de JPA en la misma máquina virtual, para proporcionar la lógica de persistencia necesaria (ver Figura 3.6).

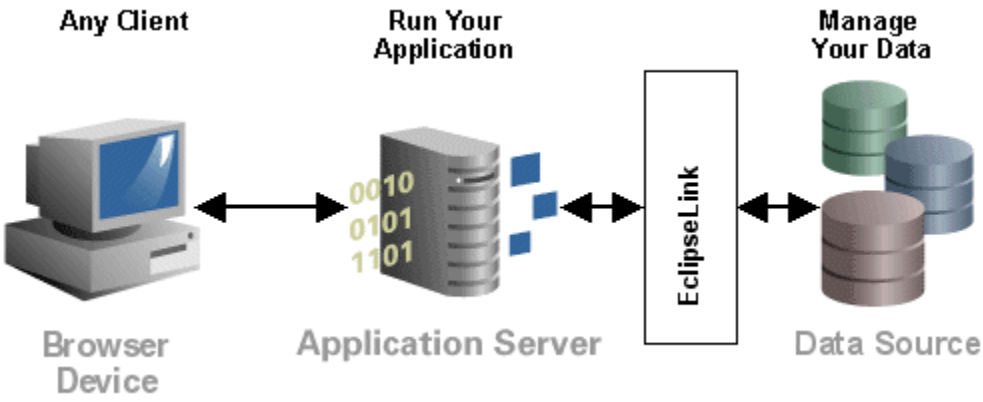


Figura 3.6 Eclipselink y una arquitectura de application general [12]

3.7. XML

XML, el Lenguaje de Marcado Extensible, es un estándar respaldado por la W3C (World Wide Web Consortium). Define una sintaxis genérica utilizada para marcar los datos con etiquetas simples y comprensibles. Proporciona un formato estándar para documentos, que es lo suficientemente flexible para ser utilizado en campos tan diversos como los sitios web, intercambio electrónico de datos, gráficos vectoriales, la genealogía, los listados de bienes raíces, la socialización de objetos, llamadas a procedimientos remotos, sistemas de correo de voz, y muchos más [13].

3.7.1. Características y beneficios de XML

En los documentos XML los datos están incluidos como cadenas de texto. Los datos están rodeados de etiquetas de texto que describen los datos. La especificación XML define la sintaxis exacta que los elementos deben seguir. Superficialmente, un documento XML luce de forma bastante parecida a un documento HTML, pero hay algunas diferencias cruciales.

Aún más importante, es que XML es un lenguaje de metamarcado. Esto significa que no tiene un conjunto fijo de etiquetas y elementos. Al contrario, XML permite a los desarrolladores y escritores a inventar los elementos que necesitan cuando lo necesitan. La X en XML significa Extensible. Significa que el lenguaje extensible puede extenderse y adaptarse para satisfacer las diferentes necesidades

Aunque XML es bastante flexible en los elementos que permite, también es bastante estricto en muchos otros aspectos. La especificación XML define una gramática para documentos XML que dice dónde pueden estar colocadas las etiquetas, como deben ser, que nombres son permitidos, y así sucesivamente. Esta gramática es lo suficientemente específica para permitir el desarrollo de analizadores XML que pueden leer cualquier documento XML. Los documentos que cumplen con esta gramática, se

dice que están bien formados. Los documentos que no están bien formados no son permitidos.

Por razones de interoperabilidad, las personas u organizaciones podrán convenir en utilizar sólo ciertas etiquetas. Estos conjuntos de etiquetas se denominan aplicaciones XML. Una aplicación XML no es una aplicación de software que utiliza XML, más bien, es una aplicación de XML en un dominio particular, tal como gráficos vectoriales o cocina. El marcado permitido en una determinada aplicación XML puede ser documentado en un esquema. Casos particulares pueden ser comparados con el esquema. Los documentos que coincidan con el esquema se dice que son válidos. Documentos que no coinciden son inválidos. La validez depende del esquema. No todos los documentos deben ser válidos. Para muchos propósitos, es suficiente que el documento esté bien formado.

Hay diferentes lenguajes de esquema XML con distintos niveles de expresividad. El lenguaje de esquema de más amplio apoyo y el único definido por la especificación XML es el DTD (Document Type Definition). Un DTD enumera todas las etiquetas legales y especifica dónde y cómo se pueden incluir en un documento. Los DTD son opcionales en XML. Por otro lado, los DTD no siempre son suficientes. La sintaxis DTD es bastante limitada y no le permiten hacer muchas declaraciones útiles, tales como "Este elemento contiene un número", o "Esta cadena de texto es una fecha entre 1974 y 2032." El Lenguaje de Esquema XML W3C permite expresar restricciones de esta naturaleza. Además de estos dos, hay muchos otros lenguajes, incluyendo RELAX NG Schematron, Hook, y Examplotron, entre otros.

- **Datos portables**

XML ofrece la tentadora posibilidad de una verdadera plataforma cruzada, en formatos de datos a largo plazo. Desde hace mucho tiempo ha sido un gran problema, el hecho de que un documento escrito en una plataforma no es necesariamente legible en una plataforma diferente, o por un programa diferente en la misma plataforma, o incluso por una versión futura o pasada del mismo programa en la misma plataforma.

XML es un formato de datos increíblemente simple y bien documentado. Los documentos XML son texto y se puede leer con cualquier herramienta que pueda leer un archivo de texto. No sólo los datos, sino también las etiquetas son texto. Se puede leer los nombres de etiquetas directamente para averiguar exactamente lo que está en el documento. Del mismo modo, el límite de cada elemento se define por las etiquetas.

XML permite que los documentos y los datos se muevan de un sistema a otro. Además, permite la validación en la parte receptora, para comprobar que se consigue lo que se espera. Java prometió código portable, XML proporciona datos portátiles. En muchos sentidos, XML es el formato de documento más portátil y flexible, diseñado desde el archivo de texto ASCII.

3.8. JSON

Según la definición de json.org, JSON es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3ra Edición - Diciembre 1999. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos [14].

JSON está constituido por dos estructuras: Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias. Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

JSON es un formato creado como una alternativa más limpia y más ligera que XML. Al igual que con XML, el objeto puede variar en complejidad desde una simple cadena a una estructura jerárquica de profundidad. También es multiplataforma. Pero en la práctica, JSON es especialmente adecuado para la comunicación con exploradores web, porque es un formato basado en JavaScript.

JSON es un formato de intercambio de datos que es un subconjunto de la notación literal de objetos en JavaScript. Se ha ido ganando mucha atención últimamente como una alternativa ligera a XML, especialmente en aplicaciones Ajax. Esto debido a la capacidad de JavaScript para analizar información de forma rápida mediante la función eval (). JSON no requiere JavaScript, sin embargo, se puede utilizar como un formato de intercambio simple para cualquier lenguaje de scripting.

3.9. Comparación entre JSON y XML

Antes de considerar las diferencias entre estos dos formatos, se listan algunas de las características que tienen en común:

- Los dos son formatos que representan un objeto como una cadena de texto.
- Al ser formatos de texto plano, ambos son adecuados para la transferencia a través de HTTP.

- Cada formato es soportado por librerías en numerosos lenguajes, incluyendo JavaScript. Hay librerías para convertir los objetos nativos de cualquier formato y volver de nuevo a los objetos nativos.

3.9.1. Ventajas de JSON sobre XML:

- JSON es más compacto, y la falta de etiquetas a veces lleva a una mejor representación visual de los datos subyacentes.
- A menudo se afirma que JSON es más rápido de analizar en los navegadores, aunque investigaciones sugieren que el análisis de XML es más escalable.
- JSON es un formato de datos concretos. XML, por el contrario, es en realidad un meta-formato, y un desarrollador tiene muchas decisiones que tomar sobre el dialecto que XML precisa para su uso. Cada uno tiene sus estrategias de asignación propias, por ejemplo, los desarrolladores necesitan decidir sobre los nombres de etiquetas y entre los atributos de la etiqueta o etiquetas anidadas.
- Dentro del navegador, JSON tiene la ventaja en términos de apoyo y consistencia, ya que está basado en el estándar de JavaScript.

3.9.2. Ventajas de XML sobre JSON:

- XML es mucho más familiar para la comunidad IT.
- XML es más auto-documentado. La cabecera identifica el formato XML se está utilizando, y hay a menudo un esquema o DTD que define el formato exacto.
- XML tiene mucho más apoyo en materia de bibliotecas y herramientas. Por otro lado, cuenta con el apoyo en términos de DTD y validadores de esquema, entre otros. Además, muchos IDEs, editores y entornos de depuración facilitan el trabajo con XML.
- Para cualquier tarea, los desarrolladores suelen tener el lujo de elegir entre varias implementaciones de la competencia.

3.10. GWT – Google Web Toolkit

Google Web Toolkit (GWT) es una potente herramienta Java creada por Google Inc. para el diseño de aplicaciones web cross-browser[6] basadas en comunicaciones AJAX.

La base del funcionamiento de esta herramienta es el GWT Compiler, un compilador que traduce código *Java a código JavaScript*. De esta manera se permite a los desarrolladores implementar la parte cliente de su aplicación utilizando alguno de los entornos Java existentes en el mercado, como Eclipse o NetBeans, agilizando así el siempre costoso desarrollo en JavaScript. Por otra parte, el compilador de GWT

[6] *Cross-browser* se refiere a la posibilidad de que un sitio web, aplicaciones web, paginas HTML o scripts del lado del cliente tengan soporte para todos los navegadores web.

permite aislar al desarrollador de los detalles y características propias de los navegadores web, haciendo posible un desarrollo multi-browser.

3.10.1. Funcionamiento de GWT

Con Google Web Toolkit, se escribe el código AJAX front-end en el lenguaje de programación Java utilizando las herramientas de desarrollo Java de su elección. Si las librerías GWT no se ajustan a sus necesidades, es posible mezclar JavaScript y Java por medio de la interfaz nativa de JavaScript (JSNI). GWT luego compila el código del lado cliente traduciéndolo a JavaScript optimizado que funciona automáticamente en todos los principales navegadores [15] (ver Figura 3.7).

Figura 3.7 Funcionamiento de GWT [15]

Utilizando GWT se puede construir una aplicación web desde cero o crear widgets únicos e integrarlos en las páginas web existentes. GWT trabaja con un navegador web especial (modo alojado) que se utiliza durante el desarrollo y depuración. Es posible recorrer el ciclo "*editar - refrescar - ver*", con la ventaja añadida de ser capaz de depurar el código Java línea por línea. En producción, el código es compilado a JavaScript, pero en modo alojado se ejecuta en la máquina virtual Java. Esto significa que cuando el código realiza una acción como la de un evento de ratón, se obtienen todas las funciones de depuración de Java, con las excepciones y las características avanzadas de depuración de IDE's como Eclipse o Netbeans.

GWT RPC[7] proporciona un mecanismo que simplifica la comunicación de una aplicación en el servidor web. Se definen las clases de Java serializables para la respectiva solicitud y respuesta. En producción, GWT serializa automáticamente la solicitud y deserializa la respuesta del servidor. El mecanismo RPC de GWT puede incluso manejar polimorfismo, y se pueden manejar excepciones normalmente. Los desarrolladores pueden diseñar y desarrollar sus aplicaciones orientadas a objetos.

Si en la parte del back-end no se desea trabajar con RPC, es posible manejar la comunicación cliente-servidor por medio de mensajes JSON o XML. Además la integración directa de GWT con JUnit permite hacer pruebas unitarias, tanto en un depurador como en un navegador, e inclusive se pueden realizar pruebas unitarias asíncronas con RPC**.**

GWT posee componentes gráficos dinámicos y reusables (*widgets*) que permiten los programadores pueden usar clases prediseñadas para implementar comportamientos que de otra manera consumirían mucho tiempo. Además maneja Internacionalización y tiene soporte para las API's de Google (inicialmente, soporte para Google Gears). Finalmente es una plataforma escalable o extensible, existe un numeroso conjunto de bibliotecas desarrolladas por Google y terceros que amplían las funcionalidades de GWT.

3.10.2. Ventajas:

GWT representa una excelente alternativa para desarrollar aplicaciones empresariales y en general cualquier aplicación web, las métricas de que utilizamos para evaluar la eficacia de GWT frente a métodos de desarrollo AJAX tradicionales son:

- **Tamaño del código JavaScript Generado.**
  Típicamente, una aplicación GWT pura requerirá que el usuario descargue en cache 100K de código JavaScript, lo que está en concordancia a lo que nos resultaría de desarrollar una aplicación AJAX a mano.

- **Rendimiento en la parte del usuario final.**
  Las aplicaciones GWT son casi siempre tan rápidas como las desarrolladas con JavaScript escrito a mano. El compilador de GWT evita la adición de *wrappers* alrededor de cualquier funcionalidad que se implementa de forma nativa en el navegador.

- **Tiempo en el desarrollo.**

[7] *El RPC* (del inglés Remote Procedure Call, Llamada a Procedimiento Remoto) es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos.

En lugar de perder tiempo resolviendo problemas que ocurren en navegadores individuales, se puede utilizar mucho más tiempo en el desarrollo de las funcionalidades de la aplicación. Esto representa una de las mayores ventajas de GWT.

Además de estas tres buenas razones para utilizar esta poderosa plataforma para desarrollar aplicaciones, se pueden mencionar otras buenas cualidades como estas:

- No se necesita mucho conocimiento de otros lenguajes (*como JavaScript*) ya que se utiliza el lenguaje de programación Java. Esto permite tener todas las ventajas de dicho lenguaje y utilizar una gran cantidad de herramientas disponibles para el mismo.
- Los errores comunes JavaScript (*types, type mismatches*) son capturados por el programador durante compilación en vez de por los usuarios durante la ejecución.
- No se necesita trabajar con las incompatibilidades de los navegadores.
- La aplicación se carga en cache, por lo que reduce el uso de ancho de banda.
- Reduce la carga en el servidor (los datos que se cargaban en la sesión del servidor ahora pasan al cliente).
- Permite la misma seguridad que otros frameworks (inclusive es más complicado inyectar datos debido al sistema de serializacion que posee).
- Contiene una gran cantidad de paneles, widgets y elementos que facilitan la parte del desarrollo de las interfaces y la parte del cliente.
- Solo es necesario editar el código Java y visualizar los cambios en el navegador en modo alojado inmediatamente sin tener que volver a compilarlo.
- GWT admite un conjunto indefinido de protocolos de transferencia, como JSON y XML, y el mecanismo de llamada a procedimiento remoto (RPC) de GWT permite el establecimiento de comunicaciones Java de una forma especialmente sencilla y eficaz.

3.10.3. Desventajas:

GWT se caracteriza por ser una plataforma robusta con varias ventajas, sin embargo al momento de escogerla como alternativa para desarrollo de una aplicación web grande y compleja, se pueden mencionar dos problemáticas fundamentalmente de carácter funcional:

- ***El tiempo que toma un ciclo de Desarrollo.***
  En modo alojado, la aplicación puede ser probada en cuestión de segundos, sin embargo al considerar la puesta en producción (despliegue) en donde se realiza la traducción de lenguaje java a JavaScript, tiene un costo en ocasiones crítico dependiendo de la complejidad de la aplicación.

- ***La curva de aprendizaje.***

Para usar GWT correctamente, es necesario utilizar una arquitectura MVC[8] y patrones de diseño cliente-servidor. Esto no suena mal para programadores experimentados, no siendo así para programadores junior para quienes la curva de aprendizaje parece ser bastante empinada.

3.10.4. Alternativas y mejoras:

GWT es un framework que como se mencionó anteriormente permite crear aplicaciones web y trabajar con el lenguaje de programación Java tanto del lado del cliente como del servidor. Esta plataforma cumple con características típicas que permiten extender o agregar funcionalidades extra como:

- ***Compatibilidad.*** Agregan la posibilidad de escribir código javascript totalmente compatible con todos los navegadores y motores Javascript más utilizados. Esto aumenta la portabilidad y eliminan el "gran dolor de cabeza" de incompatibilidad entre navegadores y sus motores intérpretes javascript.
- ***Comunicación asíncrona (Ajax).*** Usando este acercamiento, es fácil utilizar XMLHttpRequest para manejar y manipular los datos en los elementos de un sitio bien, aumentando la interactividad y experiencia del usuario.
- ***DOM.*** Maximizan la capacidad de agregar, editar, cambiar, eliminar elementos de manera dinámica agregando librerías que facilitan usar DOM.
- ***Validación de Formularios.*** Permiten de una manera relativamente fácil validar campos dentro de uno o varios formularios. Esto, desde el punto de vista del desarrollador, simplifica y reduce el código para procesar dichos formularios, ya que los datos llegan previamente validados, reduciendo los errores de tipos de datos.
- ***Efectos visuales.*** Utilizando la manipulación de los elementos, se pueden crear efectos visuales y animaciones. Entre los efectos se encuentran: aparecer y desaparecer, redimensionamiento, mover, aparecer y desaparecer y más.
- ***Almacenamiento de lado del Cliente.*** En adición provee funciones para leer y escribir cookies. También proveen una abstracción de almacenamiento que permite a las aplicaciones web guardar datos del lado del cliente de manera segura.
- ***Manejo JSON.*** Incrementa al máximo el manejo de datos, que pueden ser utilizados para presentar informaciones de manera dinámica y en tiempo de ejecución.

[8] ***Modelo Vista Controlador (MVC)*** es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos. El patrón de llamada y retorno MVC (según CMU), se ve frecuentemente en aplicaciones web, donde la *vista* es la página HTML y el código que provee de datos dinámicos a la página. El *modelo* es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el *controlador* es el responsable de recibir los eventos de entrada desde la vista.

- ***Manejo de Eventos.*** Esta característica agregada, permite reaccionar de una manera u otra dependiendo de las acciones del usuario.
- ***Recibidores de Datos.*** Permiten utilizar diferentes formatos de datos como XML, HTML, Texto, JSON, ATOM, entre otros.
- ***Funcionalidad arrastra y suelta.*** Mejor conocido como drag and drop. Es una funcionalidad que brinda la posibilidad de arrastrar elementos dentro de una misma página que interactúe con el resto de los elementos.

Algunos de los productos más conocidos de terceros que utilizan el framework de GWT y que implementan dichas características son: ***Ext GWT, SmartGWT y Vaadin.***

La extensión de GWT (ExtGWT) es una alternativa relativamente nueva, que entre otras cosas mejora el framework básico que posee GWT tanto en desempeño como en aspectos de forma (manejo de eventos, arquitectura MVC, widgets y estilos CSS). Está construido directamente usando GWT sin librerías JavaScript extra, utiliza código Java-GWT puro y un archivo extenso GXT-all.css. Por otro lado, Vaadin posee un modelo de programación muy parecido a la que poseen las aplicaciones de escritorio tradicionales, utilizando eventos y *listeners*[9] en lugar de peticiones y respuestas. Sin embargo tiene algunas limitaciones en la parte flash que utilizan sus widgets. Finalmente esta SmartGWT, que utiliza las características de *SmartClient*[10], enfocado mayormente en el desempeño y apariencia visual del lado del cliente.

Desde un punto de vista técnico Ext GWT y SmartGWT son bastante parecidos aparte del hecho de que Smart tiene más que ofrecer en el lado del servidor. Vaadin es un framework del lado del servidor que utiliza un conjunto de componentes precompilados GWT. Aunque se pueden escribir propios componentes, si es necesario. En Vaadin el cliente del navegador es sólo una visión muda de los componentes del servidor y la interacción del usuario se envía al servidor para su procesamiento al igual que los métodos tradicionales de programación web de Java. Esto constituye la principal desventaja de Vaadin, la dependencia en del servidor. Mientras que EXT GWT se pueden ejecutar en un navegador sin necesidad de comunicarse con un servidor.

Con lo dicho se hizo brevemente una revisión de cuáles son las características más relevantes de cada una de las librerías mencionadas, para tomar la decisión de cuál de ellas escoger para el desarrollo de la aplicación de microcréditos. Para esto primero se

[9] Un oyente (***Listener***) se llama cuando el usuario hace algo con la *GUI* que produce un evento.
[10] ***SmartClient*** se refiere a aplicaciones que: Son entregadas sobre la red (internet), no requieren **instalación, actualizadas automáticamente sin intervención del usuario, tienen "Look and Feel"** (aspecto) de aplicación de escritorio, Aprovechan los recursos de hardware y software del computador donde son ejecutados.

debe pensar en la complejidad que tendrá nuestra aplicación (*por cuestiones de optimización*) y que tanto conocemos de la plataforma GWT ya que todas estas son librerías basadas en dicho framework. Una buena alternativa es escoger una librería que recoja y aproveche todas las opciones, cualidades y mejoras de GWT, la mejor, desde nuestro punto de vista y enfoque de este proyecto es Ext GWT.

3.10.5. Ext GWT (Sencha GXT)

GWT permite potencializar y aprovechar aún más sus ventajas gracias a su capacidad de extensibilidad y escalabilidad, es por ello que para el desarrollo de este proyecto se utiliza una librería que como se mencionó anteriormente, cumple con los requerimientos y necesidades que demanda una aplicación web con un nivel de complejidad media-alta: Ext GWT.

Google Web Toolkit es una muy buena alternativa para que los desarrolladores Java puedan crear aplicaciones Web basadas en AJAX, sin necesidad de un profundo conocimiento de JavaScript o tener que lidiar con las peculiaridades de los diferentes navegadores. Sin embargo, provee un conjunto de herramientas en lugar de una plataforma de desarrollo completa, y para la mayoría de los proyectos, forma parte de la solución en lugar de servir como una solución completa.

Aparte de eso, GWT viene con sólo un conjunto básico de widgets[11] y carece de un marco que permita a los desarrolladores estructurar aplicaciones más grandes y complejas. Afortunadamente, GWT es una plataforma abierta y extensible y como resultado, una serie de proyectos complementarios han crecido a su alrededor. Ext GWT de Sencha es uno de esos proyectos.

Características de GXT.

Ext GWT está construido sobre las fortalezas de GWT, permitiendo a los desarrolladores dar a sus usuarios una experiencia más parecida a la de una aplicación de escritorio.

GXT ofrece al desarrollador una amplia biblioteca de componentes similares a la utilizada en el desarrollo de entornos de escritorio. Además de ser una biblioteca de componentes, proporciona potentes funciones para trabajar con datos locales y remotos. También cuenta con una arquitectura MVC, que puede ser utilizado para estructurar aplicaciones más grandes.

[11] ***Widget:*** es un elemento de una interfaz (interfaz gráfica de usuario o GUI) que muestra información con la cual el usuario puede interactuar. Por ejemplo: ventanas, cajas de texto, checkboxs, listbox, entre otros.

A continuación se lista algunas características que Sencha GXT posee:

- Utilización de Eventos y controladores de GWT.
- Soporte e interoperabilidad con Widgets GWT.
- Diseño basado en la GUI (*soporta pruebas unitarias y el patrón MVP*[12]).
- Modelo mejorado (basado en *POJO*[13]).
- Uso Plantillas.
- Campos basados en celdas y soporte para manejo de estos en los widgets.
- Nuevo y mejorado diseño para layouts.
- Soporte completo para *UiBinder*[14].
- Nueva API para gráficos y tablas estadísticas.
- Nueva Apariencia y temas.

El equipo de trabajo de Ext GWT ha estado trabajando en el lanzamiento de la versión *Ext GWT 3.0.* Esta nueva versión es una actualización con numerosos cambios, correcciones y nuevas características respecto a la versión anterior (*2.2.5*), entre las que se destaca el cambio del sistema de eventos de GXT que trabaja con el manejador de eventos de la API de GWT. Las mejoras en Ext GWT 3.0 permiten aprovechar estas nuevas funcionalidades con una curva de aprendizaje mínima. Además, la adición del nuevo plugin de gráficos, permite ampliar sus aplicaciones con la visualización de datos de gran alcance y capacidades de análisis, todo sin necesidad de plugins extra instalados en el navegador.

3.11. Resumen

En esta tercera parte se presentó un análisis general de las herramientas tecnológicas a utilizar en este proyecto. Primeramente se presentó un análisis estadístico sobre el desarrollo de aplicaciones para dispositivos móviles en el medio realizado por *Flurry* el año pasado, cuyos resultados indican que las aplicaciones nativas han ganado mayor alcance en el mercado, sin embargo las aplicaciones web siguen siendo de gran importancia especialmente cuando se requiere manejo de aplicaciones más robustas. Luego se listaron las características, ventajas y desventajas de las tecnologías utilizadas en este proyecto, tales como JPA para la parte del manejo de datos, XML y JSON para la comunicación entre la capa de datos y la parte web de cliente y finalmente se consideró Ext GWT como framework para la parte web.

[12] ***Modelo-Vista-Presentador (MVP)*** es un derivado del modelo de software modelo-vista-controlador (MVC), también se utiliza principalmente para la construcción de interfaces de usuario.
[13] Un ***POJO*** (acrónimo de Plain Old Java Object) utilizada por programadores Java para enfatizar el uso de clases simples y que no dependen de un framework en especial.
[14] ***UiBinder*** permite construir aplicaciones como páginas HTML usando los widgets de GWT.

PARTE 4
ANÁLISIS, DISEÑO E IMPLEMENTACIÓN

En los siguientes capítulos se desarrollan las tres etapas básicas del desarrollo del ciclo de vida de software para el sistema de gestión de microcréditos, bajo la metodología de diseño AUP (*Agile Unified Process*). Primero se desarrolla brevemente el tema de la metodología utilizada en este proyecto, junto con un análisis previo al diseño del sistema, que es el tema del siguiente capítulo. Finalmente se detallan las actividades que tienen que ver con la etapa de la Implementación.

Capítulo 4
Análisis del Sistema

En este capítulo se presenta el desarrollo del análisis realizado para la construcción de la Aplicación Web que servirá para la gestión de microcréditos. Como metodología de desarrollo se aplica la recomendada por *AUP (Agile Unified Process)*, misma que se explica brevemente a continuación:

4.1. Metodología de diseño AUP

El Proceso Unificado Ágil (Agile UP) describe una manera simple de entender el desarrollo de aplicaciones de negocio usando técnicas ágiles y conceptos heredados del RUP (Rational Unified Process en inglés, habitualmente resumido como RUP) de IBM. El ciclo de vida de Agile UP es *serial en lo grande* e *iterativo en lo pequeño*, liberando *entregables incrementales* en el tiempo [4].

En los proyectos que usan AUP, normalmente se entregan versiones de desarrollo al final de cada iteración. Una versión de desarrollo de una aplicación es una versión que potencialmente puede ser lanzada en producción si pasa la garantía de calidad de pre-producción, supera la fase de pruebas y los procesos de despliegue. Un enfoque en las tareas de despliegue ayuda a evitar problemas, y permite aprender de la experiencia a lo largo del desarrollo.

4.1.1. El ciclo de vida de AUP

El ciclo de vida de AUP consta de 4 fases: Inicio, Elaboración, Construcción y Transición. Además, las disciplinas en el desarrollo cambian en AUP, existen 7 disciplinas que trabajan en cada fase del ciclo de vida. Más adelante se explicará en detalle cada fase y como trabaja cada disciplina en ella. En la Figura 4.1 se ilustra la interacción entre las fases y disciplina de AUP.

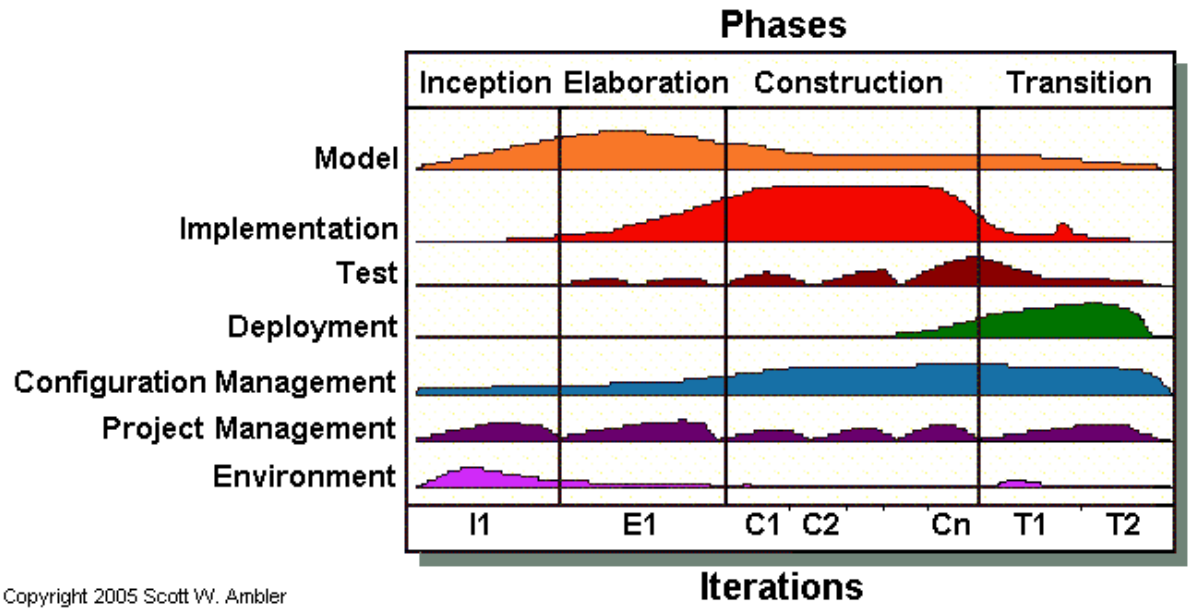


Figura 4.1 Iteraciones del Proceso de Desarrollo Ágil [4]

Fases del ciclo de vida

Agile UP está caracterizado por ser *serial en lo grande*, algo que se puede apreciar a través de estas cuatro fases las cuales se pueden mover en una forma serial:

- **Inicio**: El objetivo es identificar el ámbito inicial del proyecto, una arquitectura potencial para el sistema, y obtener financiación y la aceptación de los *stakeholders*[15].
- **Elaboración**: El objetivo es verificar la arquitectura del sistema.
- **Construcción**: El objetivo es construir un software funcional, en unas bases incrementales que cumplan con las necesidades más prioritarias del cliente.
- **Transición**: El objetivo es validar y desplegar el sistema en el entorno de producción.

Disciplinas en el ciclo de vida

Las disciplinas son ejecutadas en una forma iterativa, definiendo las actividades que el equipo de desarrollo ejecuta para construir, validar y liberar software funcional, el cual cumple con las necesidades del usuario.

[15] Estos grupos o individuos son el público interesado (”stakeholders”), que según Freeman deben ser considerados como un elemento esencial en la planificación estratégica de negocios. [18]

- **Modelado**: Abarca las disciplinas de Modelado de negocio, ingeniería de requisitos, análisis y diseño. El objetivo es entender la organización y el dominio del problema, así como encontrar una solución a éste.
- **Implementación**: El objetivo es transformar el modelo a código ejecutable y realizar pruebas de nivel básico, en particular pruebas unitarias.
- **Test**: El principal objetivo es realizar una evaluación objetiva para asegurar la calidad. Esto incluye encontrar defectos, validar que el sistema funciona como se ha diseñado y verificar que los requisitos se cumplen.
- **Despliegue**: El objetivo del despliegue es planear el desarrollo para la entrega del sistema, y ejecutar este plan para hacer que el sistema esté disponible para el usuario final.
- **Gestión de la configuración**: El objetivo es gestionar el acceso a los elementos del proyecto. Esto incluye gestionar y controlar los cambios que se hagan a estos elementos.
- **Gestión del proyecto**: El objetivo es dirigir las actividades que tienen lugar en el proyecto. Manejar riesgos, dirigir a las personas involucradas y coordinar a las personas y los sistemas involucrados para hacer que el producto esté disponible para su entrega a tiempo.
- **Gestión del entorno**: El objetivo es asegurarse de que los procesos adecuados, guías y estándares, y herramientas, están disponibles para el equipo.

4.1.2. La fase de inicio

Las actividades de la fase inicial son:

- **Definir el ámbito del proyecto**: Incluye definir en alto nivel lo que el sistema hará. Es también importante definir lo que el sistema no hará. Esto establece los límites en los que el equipo trabajará. Esto normalmente se representa mediante una lista de características o casos de uso.
- **Estimar el coste y el calendario**: Se estima en alto nivel el calendario y el coste para el proyecto. Las estimaciones generales son usadas en iteraciones en las últimas fases, más específicamente se usan en las primeras iteraciones de la fase de elaboración.
- **Definir riesgos**. Los riesgos del proyecto se definen aquí. La gestión de riesgos es una parte importante de un proyecto AUP. La lista de riesgos cambia con el tiempo, a medida que se identifican nuevos riesgos, se mitigan, se evitan y se materializan y hay que ocuparse de ellos. Los riesgos de alta prioridad se tratan en fases más tempranas que los de baja prioridad.
- **Estudio de viabilidad**. El proyecto debe tener sentido desde las perspectivas técnicas, operacionales y de negocio. En otras palabras, deberíamos ser capaces de construirlo, una vez es desplegado deberíamos ser capaces de ponerlo en marcha, y debería tener un sentido económico el hacer estas cosas. Si el proyecto no es viable, debería ser cancelado.
- **Preparar el entorno**. Esto incluye reservar un espacio de trabajo para el equipo, pedir lo que se necesite, obtener hardware y software que se necesite

inmediatamente, y crear una lista de hardware y software que necesitaremos en un futuro. Además se debe preparar el AUP para ver que necesita el equipo exactamente.

4.1.3. La fase de elaboración

El principal objetivo es probar la arquitectura del sistema a desarrollar. Hay que asegurarse de que el equipo puede desarrollar un sistema que satisfaga los requisitos, y la mejor manera de hacerlo es construir un esqueleto funcional del sistema, llamado prototipo arquitectónico.

Durante esta fase también se va preparando la siguiente. A medida que se va ganando manejo con la arquitectura del sistema, se va preparando el entorno para la construcción, adquiriendo hardware, software y herramientas.

4.1.4. La fase de construcción

El objetivo de esta fase es desarrollar el sistema hasta el punto de que esté listo para hacer pruebas de pre-producción. En las fases anteriores, la mayoría de los requisitos habían sido identificados y la arquitectura del sistema como punto de partida.

Para finalizar esta fase se debe realizar los siguientes hitos:

- Analizar y diseñar el modelo.
- Documentar las decisiones críticas de diseño.
- Construirlo.
- Ir evolucionando el dominio lógico, las interfaces de usuario y el esquema de datos.
- Probar el software.
- Desarrollar scripts para su instalación.
- Desarrollar una documentación inicial.
- Poner todos los productos bajo el CM (mantenimiento de configuración).

4.1.5. La fase de transición

Tiene como objetivo poner el sistema en producción. Se debe incrementar las pruebas en esta fase y probar una versión beta. Los hitos a realizar son los siguientes:

- Finalizar la documentación.
- Detectar fallos.
- Validar el sistema.
- Validar la documentación.
- Finalizar el modelo de prueba.

- Finalizar la documentación.
- Finalizar el desarrollo del paquete.
- Colocar el sistema en producción.
- Formar a personas.

Una vez abordado el tema del Proceso Unificado Ágil, es hora de revisar algunos de los aspectos tratados en el Análisis del Sistema. Se empieza con el análisis de riesgos que es el tópico del siguiente punto.

4.2. Análisis de Riesgos

Para el desarrollo de la aplicación se determinaron principalmente los siguientes riesgos potenciales que debieron tomarse en cuenta para llevar a cabo un diseño apropiado del sistema:

- ***Aspectos de Diseño para dispositivos móviles.***
  Dado que la aplicación es Web, es necesario crear un sistema capaz de ser ejecutado en cualquier navegador o en la mayoría de los disponibles en el mercado principalmente en dispositivos de tipo tablet, sin experimentar mayores cambios o restricciones en cuanto a la funcionalidad del mismo. El riesgo en esta parte es que existe la posibilidad de crear una aplicación restringida e inflexible. Como acción para prevenir que esto suceda, se harán pruebas funcionales durante el desarrollo, para garantizar que al final se tendrá una aplicación que evite caer en esta problemática.

- ***Manejo de Persistencia.***
  Debido a que se utilizara JPA como framework mediante EclipseLink para el manejo de la persistencia, existe el riesgo de que al ser un framework relativamente nuevo, pueda presentar limitaciones o fallas que afecten la viabilidad del proyecto. Como acción para prevenir esto se harán prototipos de forma temprana para el módulo de persistencia.

- ***Comunicación XML.***
  Como se mencionó anteriormente, la comunicación entre el módulo Web y el Core se da a través de envío y recepción de mensajes de tipo XML o JSON. GWT utiliza comúnmente RPC como mecanismo de comunicación cliente-servidor y aunque tiene soporte para el manejo de mensajes XML y JSON se desconoce la facilidad y extensibilidad de estas operaciones para manejar mensajes de complejidad arbitraria. Como acción para prevenir esta problemática se plantea poner a prueba ambas alternativas antes de elaborar completamente el módulo Core.

- ***Compatibilidad con dispositivos***

El desarrollo de esta aplicación web está orientado hacia computadores portátiles y tablets principalmente. Al existir gran variedad de proveedores, plataformas y navegadores relacionados con estos dispositivos, resulta un riesgo asociado con las posibles incompatibilidades que la aplicación pueda tener con ciertos dispositivos. Para el manejo de este riesgo se empleará Ext GWT de Sencha, que como se ha mencionado en capítulos anteriores, es una plataforma de GWT que provee muchas ventajas y que tiene compatibilidad con una cantidad respetable de navegadores.

4.3. Entorno

El Proyecto entra en la categoría de *Aplicación Web*, por lo que utiliza un navegador como cliente de ejecución y es independiente del sistema operativo donde se utilice.

Para la configuración del entorno se consideró el uso de Bases de datos locales para pruebas. El Servidor de Base de Datos a utilizar es MySQL. Para el manejo de Persistencia se consideró utilizar como herramienta EclipseLink.

Además se utilizan herramientas de trabajo en grupo como Subversion para el control de versiones y Apache Maven para la gestión y construcción de proyectos Java.

Como plataforma de desarrollo en la parte web, se utilizaron plugins del kit de desarrollo web de Google, a saber GWT y la API de mapas. Se utilizaron las librerías de Sencha Ext GWT, que permiten un manejo más completo de widgets y mejoran la experiencia de navegación con una interfaz más completa.

En la Tabla 4.1 se presenta un resumen de las herramientas utilizadas junto con las versiones correspondientes:

Herramienta	Nombre	Ver.	Descripción
Servidor Base de Datos	MySQL	5.5.8	Para el manejo de información de la Aplicación
Gestión de BD	PhpMyAdmin	3.4.5	Para administrar las BD
	DBVisualizer	7.1.4	Para administrar las BD
Lenguaje de Programación	Java	1.6	Programación de la plataforma e Interfaz de usuario
Gestión de Proyectos	Apache Maven	3	Para la Gestión de Proyectos Eclipse
Control de Versiones	Subversion	1.6	Para trabajo en grupo y control de versiones
IDE Desarrollo	Eclipse	Helios/ Indigo	Desarrollo de la Aplicación

Framework desarrollo para el front-end	GWT	2.4	Kit de desarrollo web de Google
	API Mapas	1.0.4	Librerías para el manejo de mapas con GWT
	Ext GWT	2.2.4	Extensión de GWT (Sencha Ext GWT)
Contenedor de Servlets	Apache Tomcat	7.0	Para Servidor WEB y CORE
Web Server	Jetty	7.0	Para Simulador de Core Bancario

Tabla 4.1 Herramientas utilizadas para el desarrollo del Sistema

4.4. Requisitos Funcionales de Alto Nivel

La Aplicación debe cumplir con algunos requerimientos de alto nivel para garantizar un uso apropiado de la misma y especialmente permitir crear una aplicación escalable y fácil de mantener. Como se mencionó anteriormente, deberá ser una aplicación web de carácter modular, debe permitir manejar los procesos de Negocio a través de Transacciones que se muestran al usuario como formularios y la comunicación de la parte web, o la interfaz con el usuario en general debe hacerse mediante el envío y recepción de mensajes XML o JSON. De esta manera, podrá facilitarse la comunicación con otros sistemas externos al diseñado en este proyecto. A continuación se detalla a profundidad cada requerimiento:

- ***Aplicación Modular.*** La Aplicación se encuentra organizada en Módulos que se describen a continuación:

  - Módulo Web
  - Módulo Core
  - Módulos del negocio
    - Módulo de parámetros generales
    - Módulo de seguridades
    - Módulo de personas
    - Módulo de microcrédito
  - Módulo de persistencia

- ***Transaccionalidad.*** Un sistema transaccional está diseñado para recolectar, almacenar, modificar, recuperar y en general, procesar todo tipo de información mediante operaciones individuales llamadas transacciones. Una transacción es un evento o proceso que genera o modifica la información que se encuentran eventualmente almacenada en un sistema de información.

- ***Comunicación mediante mensajes XML/JSON.*** El usuario puede utilizar la aplicación y manipular la información por medio de formularios (transacciones) y esta información es enviada a procesar por medio de mensajes de tipo XML o JSON, dichos mensajes permiten una comunicación cliente servidor más rápida y versátil además permite interactuar con otras aplicaciones que utilicen un esquema de comunicación similar.

- ***Manejo de un flujo de procesos.*** Los procesos que ejecutará la aplicación están estructurados según los módulos de Negocio. Así por ejemplo, el *Módulo de parámetros generales* contiene los procesos necesarios para paramétrizar información que será utilizada por otros módulos de la aplicación, también contiene la parametrización de localizaciones: países, provincias, cantones, parroquias. La estructura del flujo de procesos completa se detalla en el apartado 5.7.3 del siguiente capítulo.

- ***Interconexión con un Simulador de Core Bancario.***
  El sistema a desarrollar en esta tesis pretende cubrir los procesos de planificación, comercialización, seguimiento y recuperación de microcréditos, más no los procesos de contabilidad, análisis, procesamiento y manejo en general del área de colocaciones de una entidad financiera. Estos procesos tienen una complejidad sumamente alta y son efectuados en sistemas conocidos como Core Bancarios. En virtud de ser un sistema aislado a un core bancario se puede manejar por separado. Sin embargo, es necesario que el sistema esté en capacidad de interactuar con un sistema de estas características. Puesto que, en la práctica los procesos de análisis, calificación, instrumentación, contabilización y maduración de los microcréditos son realizados en un core bancario. De allí que, surge la necesidad de desarrollar un Simulador o Emulador de core bancario, que será una implementación aislada que hará las veces de un core bancario. Y que se utilizará principalmente en el proceso de instrumentación (creación de cuotas a partir de los datos de la solicitud) de los microcréditos.

4.5. Requisitos no Funcionales

Algunos de los requerimientos no funcionales a tomar en cuenta son:

- ***Desempeño***
  - El sistema debe presentarse funcional en varios navegadores: al menos en Chrome, Firefox, e Internet Explorer para aplicaciones ejecutadas desde PC's o laptops. Y debe ser adecuada para ejecutarse en navegadores de dispositivos móviles como tablets, principalmente en el ipad.

- El sistema debe estar en capacidad de dar respuesta al acceso de todos los procesos con tiempo de respuesta aceptable y uniforme, en períodos de alta, media y baja demanda de uso del sistema.

- **Escalabilidad:**
  - El sistema debe ser construido sobre la base de un desarrollo evolutivo e incremental, de manera tal que nuevas funcionalidades y requerimientos relacionados puedan ser incorporados afectando el código existente de la menor manera posible; para ello deben incorporarse aspectos de reutilización de componentes.
  - El sistema debe estar en capacidad de permitir en el futuro el desarrollo de nuevas funcionalidades, modificar o eliminar funcionalidades después de su construcción y puesta en marcha inicial.

- **Facilidad de Uso e Ingreso de Información:**
  - El sistema debe ser de fácil uso y entrenamiento por parte de los usuarios (Asesores).
  - El ingreso de información al sistema debe diseñarse con transacciones que permitan el ingreso de los datos de forma sencilla, considerando aspectos de diseño para dispositivos tablets.
  - El sistema debe presentar mensajes de error que permitan al usuario identificar el tipo de error y comunicarse con el administrador del sistema.

- **Facilidad de mantenimiento:**
  - El sistema debe contar con una interfaz de administración que incluya: Administración de usuarios, Administración de subsistemas, módulos y procesos; así también administración de parámetros y datos generales. En cada una de éstas secciones deberá ofrecer todas las opciones de administración (mantenimiento) disponibles para cada uno.

- **Seguridad:**
  - El acceso al Sistema debe estar restringido por el uso de claves asignadas a cada uno de los usuarios. Sólo podrán ingresar al Sistema las personas que estén registradas, estos usuarios serán clasificados según roles con acceso a las opciones de trabajo definidas para cada rol.
  - El control de acceso implementado debe permitir asignar los perfiles para cada uno de los roles identificados.

- **Validación de Información**
  - El sistema debe validar automáticamente la información contenida en los formularios de ingreso. En el proceso de validación de la información, se deben tener en cuenta aspectos tales como obligatoriedad de campos, longitud de caracteres permitida por campo, manejo de tipos de datos, etc.

- **Arquitectura:**
  - La solución debe ser 100% Web Based y toda la parametrización y administración debe realizarse desde un navegador.
  - La solución debe operar de manera independiente del navegador que se utilice.
  - Debe tener interfaces gráficas de administración y de operación en idioma español y en ambiente 100% Web, para permitir su utilización a través de exploradores o navegadores de Internet.
  - La información de los formularios que corresponda a listas de selección deberá ser parametrizada y administrable.

4.6. Resumen

En este capítulo se trató la etapa de diseño del proceso de desarrollo de la aplicación para la gestión de microcréditos. Para esto, se inició mencionando el Proceso Unificado Ágil (AUP), que es la metodología empleada. La metodología AUP es una metodología serial en lo grande e iterativo en lo pequeño, y es muy utilizada por su característica de ir desarrollando entregables al final de cada iteración. Se presentó también un análisis de riesgos, en donde se prioriza los riesgos según su importancia. Luego se describió asuntos relacionados con la organización previa del entorno de desarrollo, en donde se destaca el uso de herramientas libres. Finalmente, se analizó los requisitos funcionales y no funcionales a tener en cuenta para el desarrollo del sistema.

Capítulo 5
Diseño del Sistema

Para abordar el diseño del sistema se parte de un enfoque global en el que se analiza la arquitectura del sistema, la comunicación entre los módulos y el modelo MVC empleado en el diseño de la interfaz gráfica de usuario. Posteriormente se profundiza en la organización de los procesos de la gestión de microcrédito, con lo que se da paso al diagrama de casos de uso y la estructura del mensaje a utilizar. Para finalmente, graficar algunos de los diagramas más importantes de la etapa de diseño: diagrama de componentes, de clases, entidad-relación, de secuencia y de despliegue.

5.1. Arquitectura del Sistema

La arquitectura modular planteada para el sistema, hace que exista una separación fuerte entre la parte web y la parte del core. La parte web se encarga únicamente de la funcionalidad básica del front-end. Mientras que la parte del core es la encargada de todo el procesamiento central. La única forma de comunicación entre el core y la parte web es por medio del intercambio de mensajes JSON o XML.

Por otro lado para la parte web se usa la arquitectura MVC (Modelo Vista Controlador, Model View Controller) que tiene soporte en Ext GWT. El patrón MVC aumenta las capacidades de la arquitectura conocida como Modelo Cliente/Servidor y a su vez organiza todos los componentes que conforman al sistema según su función. Al implementar el sistema siguiendo la arquitectura mencionada anteriormente se obtienen todas las características más utilizadas en la actualidad para el desarrollo de aplicaciones web y las ventajas de los sistemas distribuidos. En la Figura 5.1 se muestra un gráfico que ilustra la arquitectura y el funcionamiento básico del sistema

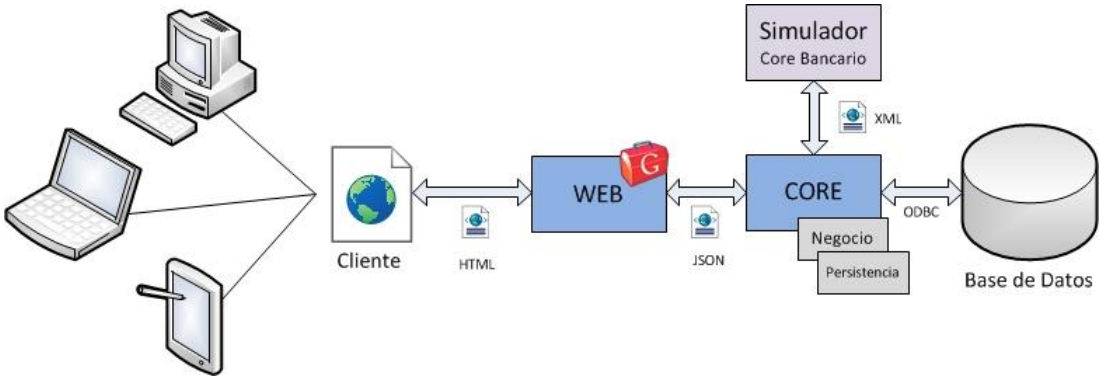


Figura 5.1 Arquitectura del Sistema

Como se mencionó en capítulos anteriores y como se observa en la Figura 5.1 el sistema está dividido en tres módulos principales: Web (cliente), Core (procesamiento y negocio) y Persistencia (acceso a base de datos). La plataforma que se utiliza en el proyecto para el módulo Web es Google GWT con la extensión de Sencha Ext GWT y para la Persistencia es JPA con EclipseLink. En la Parte del Servidor se hace uso de Java Servlets que son los encargados de recibir los requerimientos, procesarlos y devolver la respuesta al cliente. Y para la comunicación con el Core se emplea la transmisión de mensajes JSON o XML.

El proceso comienza cuando el cliente accede a la página principal de la aplicación y realiza alguna acción (ejemplo el usuario ingresa información en un formulario). En esta parte el módulo Web se encarga de recoger los datos ingresados por el usuario y enviarlos en un mensaje hacia el Core, donde el mensaje es procesado. En el módulo Core se ejecutan los procesos de negocio sobre el mensaje, y se devuelve hacia la parte Web con los datos procesados y un código de respuesta además de códigos de error, en caso de haber existido alguno. Finalmente, el módulo Core ofrece la posibilidad de comunicarse con un servlet externo para realizar operaciones de procesamiento externo. Este es el caso de la comunicación que se puede establecer entre el Core y el Simulador de Core Bancario, para realizar la instrumentación (creación de cuotas a partir de los datos de la solicitud de microcrédito) de los préstamos realizados en el proceso de venta de un microcrédito.

5.2. Comunicación Web-Core-Persistencia

Una diferencia fundamental entre Google Web Toolkit y las tradicionales aplicaciones web, es que las aplicaciones GWT no necesitan de otras páginas web mientras son ejecutadas. Ya que las páginas construidas con este framework corren como aplicaciones sobre el navegador. Por lo tanto, no necesitan hacer nuevas peticiones al servidor para, por ejemplo, realizar actualizaciones en la interfaz de usuario. Sin embargo, como todas las aplicaciones cliente/servidor, los programas en Google Web Toolkit necesitarán pedir ciertos datos al servidor para realizar determinadas tareas.

Existen dos mecanismos que se utilizan comúnmente para interactuar con el servidor a través de la red, el primero es llamado “Remote Procedure Call” (RPC), que viene siendo en español “Llamada a Procedimiento Remoto”. El RPC en GWT permite fácilmente al cliente enviar y recibir objetos de Java sobre HTTP. Otra forma de comunicación con el servidor es a través de archivos de tipo JSON o XML, GWT puede utilizar estos mensajes para extraer los datos de servidores remotos. Los datos se pueden transmitir en ambas direcciones. JSON se considera menos detallado, pesado y legible que XML, aunque estas cualidades podrían variar con los datos transmitidos.

La arquitectura de la aplicación permitirá el intercambio de mensajes de tipo XML o JSON, según se requiera. El uso de mensajes JSON es recomendado para el caso de

aplicaciones web, y por esa razón la comunicación entre la parte Web y el Core se hace mediante mensajes JSON. Por otro lado, para la comunicación entre el Core y el Simulador del core bancario se utiliza mensajes XML, esto para probar el manejo de ambos tipos de mensaje en el sistema.

5.3. Modelo MVC

Como se ha mencionado anteriormente, para la implementación del módulo Web se hará uso de las librerías de Sencha Ext GWT (GXT). Para un mejor manejo de eventos y control sobre los datos, se utiliza el patrón MVC. Éste patrón en GXT es un poco diferente del patrón MVC convencional, pero sigue siendo muy útil. En la Figura 5.2 se resume brevemente el funcionamiento de este modelo.

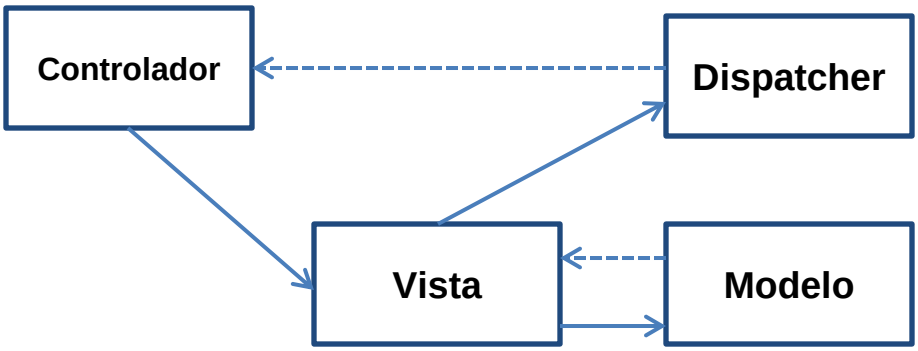


Figura 5.2 Modelo Vista Controlador de GXT

A continuación se describe el funcionamiento de cada una de los componentes de la Figura 5.2:

- **Modelo:** Toma forma de objetos de tipo ModelData, que es una interfaz para el manejo de datos basada en un mapa de datos. Los objetos individuales pueden ser recuperados y manipulados a través de sus métodos get y set.

- **Vista:** Organiza los componentes de la interfaz de usuario. Al igual que con el clásico patrón MVC, está a la escucha y responde ante de los cambios en el modelo. A diferencia del clásico MVC, los componentes de la vista pueden hacer cambios en el modelo mediante la carga de los datos en él. La vista utiliza el despachador (dispatcher) para disparar eventos que son observados por el controlador.

- **Controlador:** El patrón clásico MVC responde a los eventos recibidos desde la vista a través del dispatcher. Entonces se puede realizar tanto una operación en el modelo o disparar un evento hacia la vista.

- **Dispatcher:** en lugar que el controlador tenga comunicación directa con la vista, la vista está a la escucha de los eventos disparados por el dispatcher. El Dispatcher es una clase con métodos estáticos que pueden ser llamados para reenviar eventos a los controladores. El controlador entonces se comunica con el dispatcher para recibir tipos específicos de eventos [16].

Para el front-end las vistas principales serán básicamente cuatro: la vista de la aplicación, la vista del menú principal, la vista de formularios o procesos y la vista de las notificaciones. Estas vistas están relacionadas con los componentes principales del front-end.

- **Ventana principal (Vista de la aplicación).** Está compuesta principalmente del encabezado, menú principal, área de formularios y área de notificaciones. Se encarga de los eventos generales tales como inicialización de la interfaz, presentación de errores y cierre de la sesión.

- **El árbol de procesos (Vista del menú principal).** Es un panel que contiene una estructura donde se muestran todos los procesos de la aplicación, ordenados jerárquicamente por subsistemas, módulos y procesos. Los eventos principales que maneja son el de selección de proceso y cambio de proceso.

- **Área de formularios (Vista de formularios).** Representa el área de trabajo principal donde se visualizan los formularios de la aplicación de microcrédito. Al igual que el menú principal los eventos que maneja son el de selección y cambio de proceso. Pero, además maneja el evento de recarga de formulario, que hace que se vuelva a cargar desde cero el formulario actual.

- **Panel de notificación (Vista de notificaciones).** Cuando se realiza alguna acción, la aplicación muestra notificaciones ya sean de errores de validación o de estado en general, que le permiten al usuario ir verificando como se ejecuta la transacción durante su utilización. Esta es la tarea del área de notificaciones ubicada en la parte inferior de la interfaz. El evento que se dispara es el de visualización de mensaje.

5.4. Flujo de Procesos del Negocio

A continuación se menciona el flujo de procesos involucrados en la gestión de microcréditos, que constituye el tema principal de la lógica de negocio que se aplicará

sobre la arquitectura de la que se hablado en secciones anteriores de este capítulo. El flujo de procesos se encuentra descrito gráficamente en la Figura 5.3. Se identifican dos escenarios principales: el de configuración inicial y el flujo de procesos de negocio.

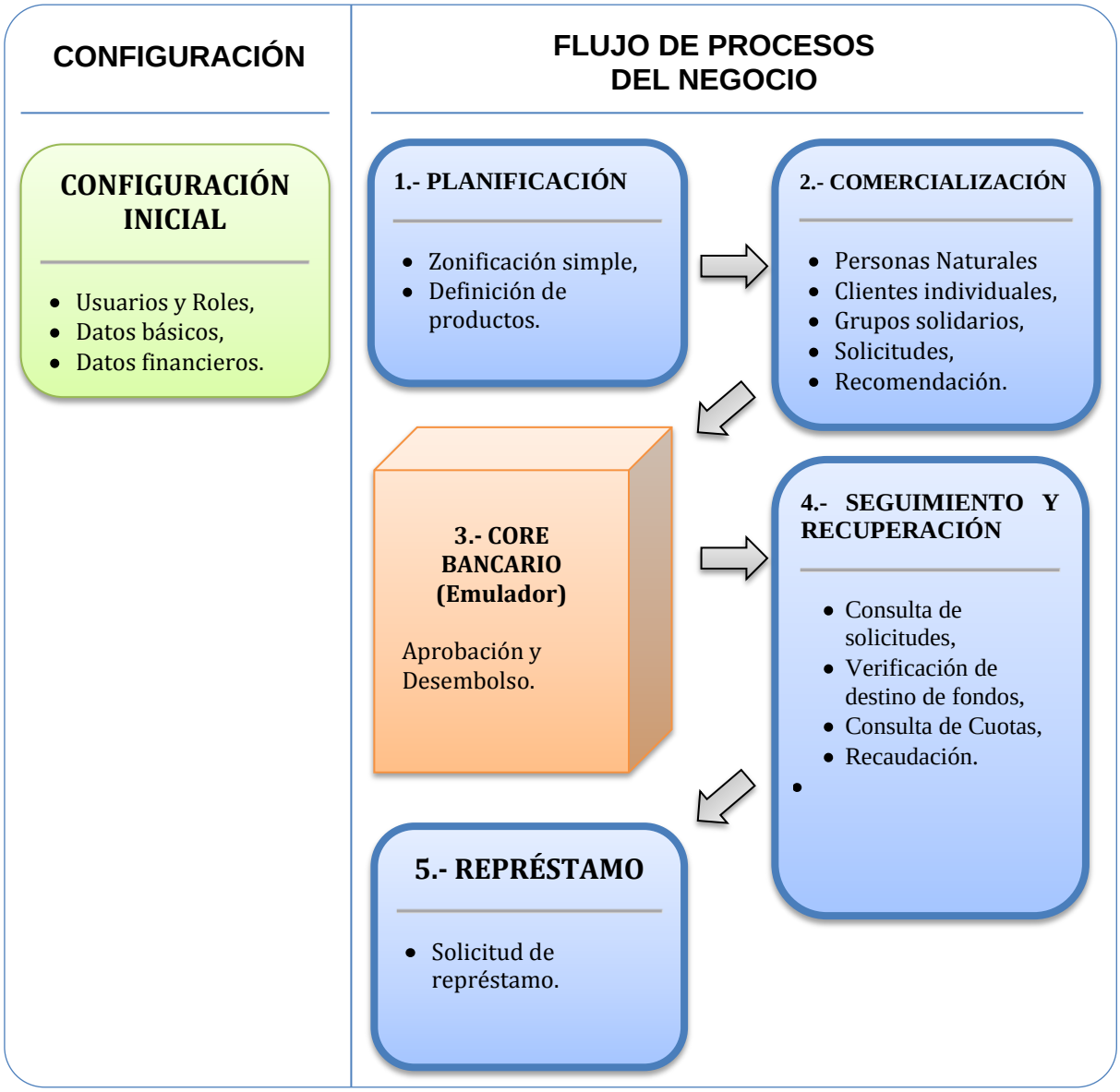


Figura 5.3 Flujo de procesos para la gestión de microcréditos

- **Configuración Inicial.** Trata los procesos relacionados con las configuraciones iniciales. No está relacionado directamente con los procesos de microcrédito. Sin embargo, estos procesos son necesarios porque manejan la información relacionada con la seguridad y parámetros generales de la aplicación. A continuación se describe los módulos relacionados con esta etapa

- **Módulo de seguridad:** Se encarga del manejo (creación, actualización y eliminación) de usuarios, roles, y se definen autorizaciones, procesos, etc. En general, un manejo simple de procesos de seguridad.
- **Parametrización:** Se refiere a la creación de los scripts SQL para el ingreso de los datos básicos para las tablas de la BD que lo requieran. Ejemplo: Parametrización de tablas de países, provincias, estados, tipos, etc. Además del manejo de los parámetros generales de la aplicación.

- **Gestión de microcréditos.**

  En este escenario se encuentran los procesos directamente relacionados con la gestión de microcrédito. Dichos procesos van desde la definición de zonas y productos crediticios hasta la recuperación de los mismos. Se divide en:

  - **Etapa de Planificación:** Aquí se definen los procesos que se realizan de forma previa a la comercialización de microcréditos. Los mismos que son realizados generalmente por el Coordinador de Microcrédito, e involucra tareas como la especificación de los lugares donde los asesores deben ir a comercializar los productos crediticios. La planificación se ejecuta bajo la responsabilidad de los directivos o encargados de la entidad Bancaria a través de reuniones, análisis estadísticos y poblacionales, etc. En el alcance de este se requiere la implementación de únicamente dos módulos:

    - **Módulo de planificación:** encargado de la asignación de determinados puntos geográficos a usuarios (Asesores) mediante utilización de mapas. El Coordinador es el responsable de asignar las zonas a los asesores y estos al acceder al sistema verifican las mismas.

    - **Módulo de productos crediticios:** para el manejo de los diferentes productos de microcrédito, en donde se deberá especificar parámetros tales como: montos, forma y frecuencia de pago, tasas, costos, etc.

  - **Etapa de Comercialización:** Es en esta etapa en la que el asesor de microcrédito toma parte importante en la gestión de una unidad de microcrédito. Se consideran los siguientes módulos:

    - **Módulo de personas Naturales:** aquí se definen los datos de los clientes o socios que aplican para un préstamo, es decir datos personales, teléfonos, direcciones, etc.

- **Módulo de clientes individuales:** para el manejo de la información de las personas que aplican para la solicitud del microcrédito de manera individual. Se incluye información del cliente y datos adicionales tales como: el asesor responsable de dar seguimiento a la solicitud, frecuencia de visita, día de reunión y actividades a las que se dedica el socio.

- **Módulo de clientes grupales:** permite el manejo de los grupos solidarios. En primer lugar se debe crear un grupo con información de asesor responsable, frecuencia de visita, día de reunión y actividades que realizan sus miembros, después se debe vincular cada grupo con las personas que lo integran. A saber, suelen ser de 3 a 15 socios y cada uno debe tener un nivel de responsabilidad en el mismo. Esto mediante el nombramiento de un presidente, un secretario y los demás definidos como integrantes.

- **Módulo de solicitud y aprobación:** permite el ingreso y actualización de solicitudes de microcrédito que contendrán la información recopilada en los módulos anteriores y datos principalmente de carácter financiero tales como montos de préstamo, plazos, frecuencia de pago, destino de fondos, etc. En este módulo también se trata el tema de los représtamos. Dado que se maneja un esquema de aprobación directa, en este formulario se puede aprobar o rechazar directamente una solicitud. La siguiente etapa, una vez creada y aprobada la solicitud es la instrumentación, que es el tema del siguiente módulo. También se puede crear recomendaciones, que son registros descriptivos sobre el cliente para una solicitud dada.

- **Etapa de instrumentación, seguimiento y recuperación.** En esta etapa se realiza la instrumentación de los microcréditos solicitados en la etapa anterior. Luego siguen los procesos de seguimiento y recuperación que serían los pasos finales previos a la opción del représtamo. A continuación una descripción de cada módulo de esta etapa.

  - **Módulo de instrumentación:** en esta parte se realiza el proceso de instrumentación de las solicitudes aprobadas por el asesor. La instrumentación de un microcrédito en forma general consiste en la ejecución de los procesos contables relacionados con la apertura de una cuenta de préstamo junto con la creación de las cuotas que son calculadas a partir de los datos de la solicitud. Este proceso se realiza en conjunto con el Simulador de Core Bancario, que es donde se simula el proceso de instrumentación. En un entorno real

la instrumentación se ejecutaría en el Core Bancario de la entidad financiera.

- **Módulo de seguimiento y recuperación:** permite el manejo y ejecución de los pagos periódicos, recolección de los datos asociados al seguimiento de los proyectos de microcrédito. Estas tareas serán realizadas por el asesor del microcrédito en las reuniones periódicas que deberá mantener con los clientes. Incluye los procesos de verificación de destino de fondos además de la consulta y pago de cuotas.

5.5. Diagrama de Casos de Uso

A continuación, en la Figura 5.4 se muestra el diagrama de casos de uso para dos de los usuarios principales del sistema: el Coordinador y el Asesor de microcréditos.

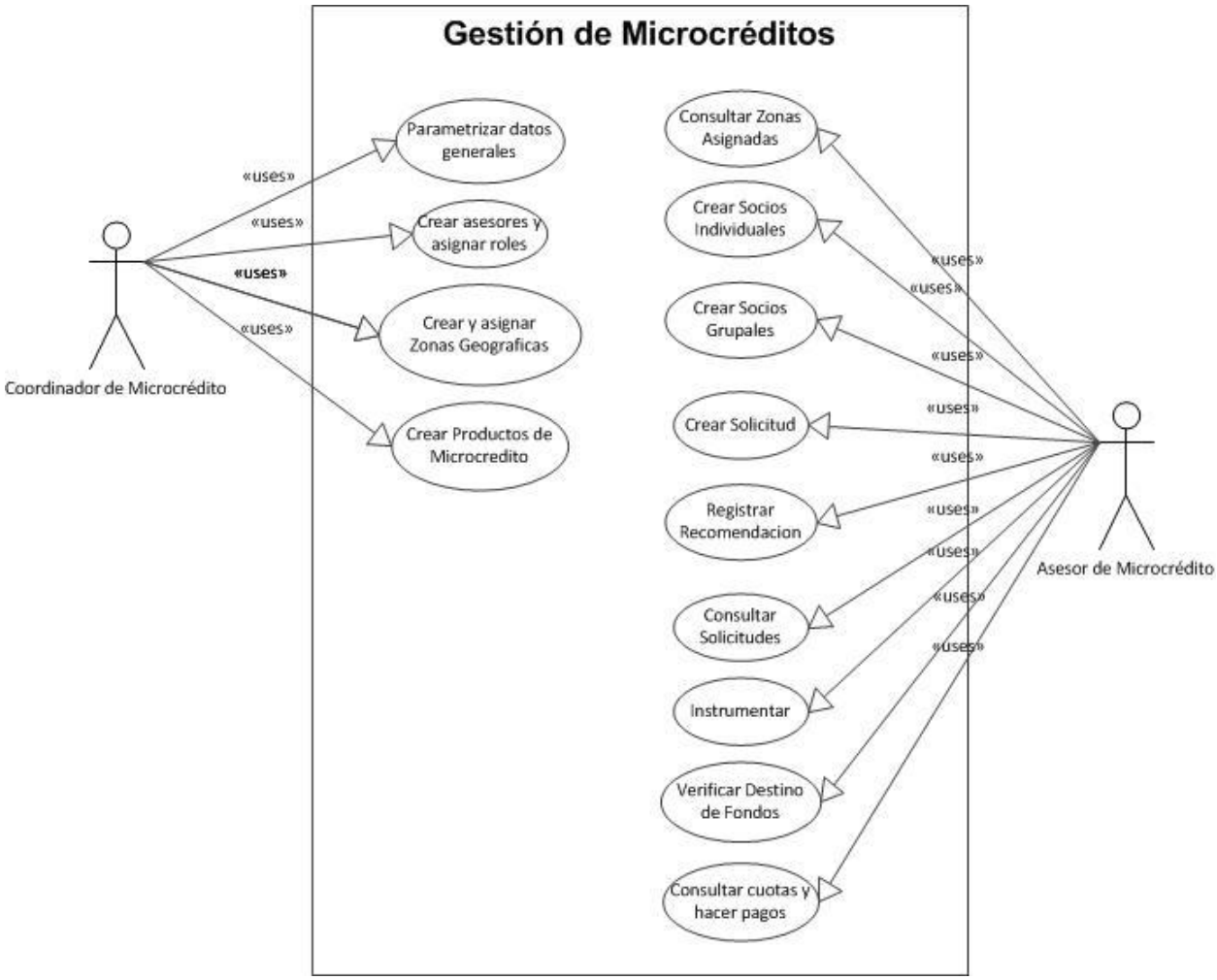


Figura 5.4 Diagrama de casos de uso general

En el siguiente punto se detallará los casos de uso de la Figura 5.4.

5.6. Detalle de Casos de Uso

En el diagrama de la Figura 5.4 se describe algunos de los casos de uso más importantes del sistema. A continuación se detallará algunos de ellos, mientras que para una referencia completa de los casos de uso, se incluirán en el Anexo 1.

Nombre:	***Acceder al Sistema***
Actores:	Coordinador de Microcrédito, Asesor de Microcrédito
Función:	Permitir acceso al sistema

Descripción:	Existen Usuarios registrados, solo estos tienen acceso al sistema. Se presenta pantalla de Inicio de sesión. Si el acceso es exitoso muestra ventana de selección de rol. Luego se carga la ventana de inicio y el árbol de procesos según el rol escogido.

Nombre:	***Crear Usuarios***
Actores:	Coordinador de Microcrédito
Función:	Proceso de creación de Usuarios
Descripción:	El Administrador es el responsable de crear nuevos usuarios, generalmente de tipo Asesor. Debe existir un script SQL que parametrice usuarios iniciales.

Nombre:	***Crear Productos de Microcrédito***
Actores:	Coordinador de Microcrédito
Función:	Ingreso de Productos a ser comercializados
Descripción:	Se definen los productos a ser comercializados, sus montos, periodos mínimos y máximos en que se realizan los préstamos y la tasa de interés definida según especifica El Banco Central del Ecuador. Una vez creados los productos se deben asignar a los asesores para que estos tengan acceso a su promoción en etapas posteriores.

Nombre:	***Crear Zonas Geográficas***
Actores:	Coordinador de Microcrédito
Función:	Creación y modificación de Zonas Geográficas
Descripción:	El coordinador puede ingresar zonas geográficas a través de coordenadas en un mapa. Las zonas pueden ser de tres tipos: puntos, rutas y polígonos. Es posible modificar las zonas en caso de que el coordinador requiera hacerlo.

Nombre:	***Crear Socios Individuales***

Actores:	Asesor de Microcrédito
Función:	Ingreso y modificación de socios individuales.
Descripción:	El Asesor puede crear socios o clientes individuales por medio de un formulario donde debe especificar: Datos de la persona, nombre del asesor responsable del socio, la frecuencia de reunión, el día de reunión y las actividades a las que se dedica el socio de manera descriptiva.

Nombre:	***Crear Solicitud***
Actores:	Asesor de Microcrédito
Función:	Creación de Solicitud de microcrédito
Descripción:	El Asesor coloca la información recopilada en los módulos anteriores sobre datos del cliente, del asesor que ingresa la solicitud y datos principalmente de carácter financiero tales como montos de préstamo, plazos, frecuencia de pago y destino de fondos. La Aprobación es inmediata y es responsabilidad del Asesor, por lo tanto en este mismo proceso se coloca el estado de la solicitud.

5.7. Estructura de Procesos

La Aplicación según reglas de negocio sugiere disponer de una estructura de tipo jerárquico. Dado que ésta es de carácter transaccional, su organización básicamente se encuentra establecida de la siguiente manera: Subsistemas, Módulos y Procesos.

5.7.1. Subsistemas.

Representan el nivel más alto en la Jerarquía del modelo de procesos que se ejecutan para la gestión de microcréditos, su unidad de representación son las letras del abecedario, por ejemplo: A, B, C…Z. Un subsistema representa a los módulos de negocio considerados para esta aplicación, es decir aquí se definen los Subsistemas de Generales, Seguridades, Personas, y Microcrédito (Ver Tabla 5.1).

Subsistema	Nomenclatura	Descripción
Generales	G	Parámetros generales de sistema y para datos de Localización.
Seguridades	A	Para transacciones que tienen que ver con la seguridad del sistema, registro de usuarios, roles, etc.

Personas	B	Para el ingreso de parámetros, datos personales, direcciones y teléfonos de personas que serán los socios de microcrédito o usuarios del sistema.
Microcrédito	C	Para la gestión de los procesos de microcrédito.

Tabla 5.1 Subsistemas de la Aplicación

5.7.2. Módulos.

Los módulos representan el segundo nivel en la jerarquía de la estructura del negocio, se representa con numeración de un dígito y hace referencia a los grupos procesos que se llevan a cabo dentro de cada subsistema, a continuación en la Tabla 5.2 se muestra en detalle módulos de cada subsistema, junto con una breve descripción de los procesos que agrupan.

Sub.	Mod.	Título	Descripción
G	0	Menú	Procesos relacionados con el menú principal.
G	1	Parámetros	Manejo de parámetros del sistema
G	2	Listas de valores (Combos)	Para listas de valores que requieren consultas especiales.
G	3	Localización	Manejo de datos de localización, como son Países, Provincias, Cantones y Parroquias
B	0	Parametrización	Parametrización de datos generales necesarios para el manejo de personas.
B	1	Personas naturales	Ingreso de personas naturales, así como direcciones y teléfonos.
A	0	Autenticación	Procesos de ingreso y salida del sistema.
A	1	Datos generales	Parametrización de datos generales del subsistema de seguridad.
A	2	Usuarios y Roles	Manejo de usuarios, roles, asignación de roles a usuarios y especificación de procesos por rol. Además de procesos de cambio de reseteo de contraseñas y registro de terminales.
C	0	Parametrización	Parametrización de datos generales del subsistema de microcréditos.
C	1	Planificación	Contiene procesos relacionados con la etapa de planificación de microcréditos. Aquí se definen asesores, zonas geográficas y productos de microcrédito.

C	2	Comercialización	La comercialización se refiere al ingreso de clientes individuales y grupales.
C	3	Solicitud	Es acorde a la etapa en donde se crea la solicitud de microcrédito, y se puede crear recomendaciones también.
C	4	Instrumentación Core	Contempla los procesos encargados de la instrumentación con el Simulador de Core Bancario.
C	5	Seguimiento y recuperación	Tiene los procesos de verificaciones y cobro de cuotas.

Tabla 5.2 Módulos de la Aplicación

5.7.3. Procesos.

Se encuentran en el tercer nivel en la estructura del negocio, se representan con dos dígitos numéricos y en la aplicación son la unidad de transacción que se ejecuta, procesa y permite la manipulación de la información por parte del usuario. En otras palabras son los formularios que el usuario ve y utiliza para realizar cada una de las operaciones del sistema. Para esta aplicación se han definido una serie procesos que por su cantidad se han incluido en el Anexo 2. A continuación en la Tabla 5.3 se muestra los procesos pertenecientes al subsistema de microcréditos.

Sub.	Mod.	Pro.	Título	Descripción
C	0	-	Parametrización	
C	0	01	Monedas	Manejo de monedas
C	0	02	Estatus de solicitud	Manejo de estatus de solicitud
C	0	03	Tipos de Cuota	Manejo de tipos de cuota
C	0	04	Frecuencias	Manejo de frecuencias (para visitas y pagos)
C	0	05	Destinos de fondos	Manejo de destinos de fondos
C	1	-	Planificación	
C	1	01	Asesor de Microcrédito	Listado de Asesores registrados en el sistema
C	1	02	Zonas Geográficas	Mantenimiento de zonas con manejo de mapas
C	1	03	Zonas por Asesor	Asignación de zonas geográficas a cada asesor
C	1	04	Definición de productos de microcrédito	Creación de productos de microcrédito

C	1	06	Zonas Asignadas	Accesible solo por los Asesores para visualizar las zonas que le fueron asignadas
C	2	-	Comercialización	
C	2	01	Clientes individuales	Creación de clientes Individuales
C	2	02	Clientes grupales	Creación de grupos y asignación de miembros a estos.
C	3	-	Solicitud	
C	3	01	Solicitud de microcrédito / Représtamo.	Ingreso de Solicitudes. Aprobación/Denegación inmediata
C	3	02	Recomendación	Proceso de registro de Recomendación dada por el asesor
C	4	-	Instrumentación Core	
C	4	01	Consulta de solicitudes	Mostrar un listado de las solicitudes y sus estados. Útil previo a la instrumentación.
C	4	02	Instrumentación en lote	Instrumentación empleando el Simulador de Core Bancario.
C	5	-	Seguimiento y recuperación	
C	5	01	Verificación de destinos de fondos	Proceso para verificación de destino de fondos
C	5	02	Consulta de cuotas	Listado de cuotas según número de solicitud
C	5	03	Pago de cuotas	Listado de cuotas con opción de pago

Tabla 5.3 Procesos para el Subsistema de Microcrédito

5.8. Estructura del Mensaje

El mensaje es el medio de comunicación que utilizan las transacciones (procesos) del sistema. Una transacción en términos generales se manifiesta por el envío y recepción de un mensaje, donde el mensaje de ida contiene los datos de entrada y el mensaje de salida, devuelve la información procesada. El mensaje puede estar estructurado en formato XML o JSON para su transmisión.

La estructura del mensaje tiene 4 partes principales: *La cabecera, datos de trabajo o entidades, datos de control y datos de respuesta.* En la Figura 5.5 se muestra una representación gráfica de las partes del mensaje.

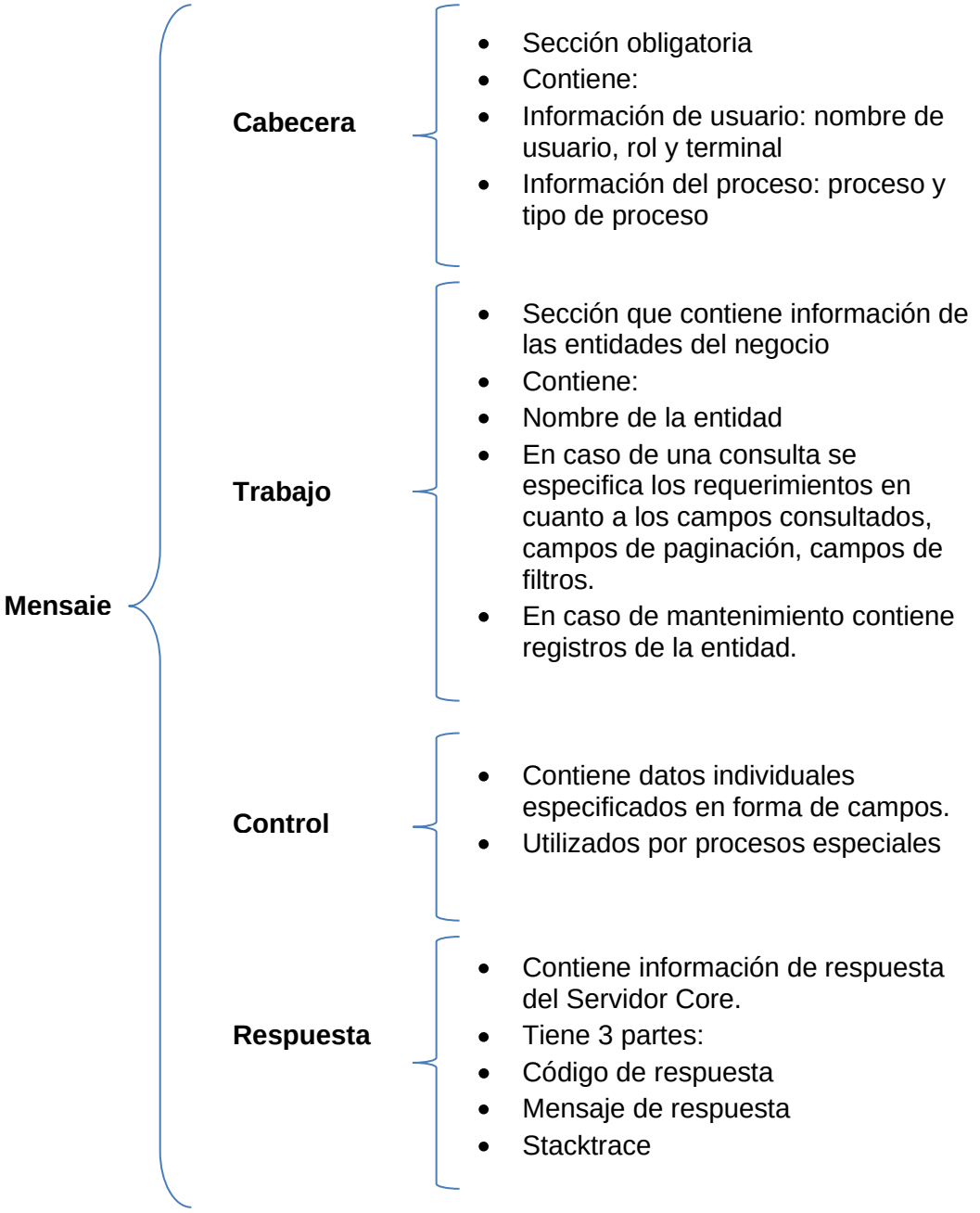


Figura 5.5 Estructura del mensaje

Cada una de las 4 partes del mensaje tiene una funcionalidad diferente. A continuación se describe brevemente cada una de estas partes:

5.8.1. La Cabecera.

Esta sección es absolutamente necesaria y por lo tanto obligatoria. En ella se especifica información básica que el Servidor Core necesita para coordinar su procesamiento. En primer lugar se especifica la información del usuario que ejecuta la transacción. De esta forma contiene el nombre del usuario, el rol y el terminal desde donde se encuentra registrado. El nombre de usuario y el rol son utilizados con fines de seguridad y validación de permisos. Un usuario sólo podrá realizar transacciones que están registradas en su rol. Además de estos datos la cabecera posee información referente al proceso que se está ejecutando. Por lo tanto tiene el identificador del proceso (ver sección 5.7) y tipo de proceso. El tipo de proceso se emplea para diferenciar entre un mantenimiento (MNT - Maintenance) y una consulta (QRY - Query).

5.8.2. Datos de Trabajo.

En esta sección se especifican los datos de las Entidades (Entidades persistentes de la base de datos) con las que se trabaja ya sea para la consulta o el mantenimiento. Permite establecer un alias, como identificador alterno para la entidad.

En caso de una transacción de consulta se especifican otros datos como los campos que se requiere consultar, los campos de paginación y los filtros que se debe aplicar en la consulta.

En caso de un mantenimiento, la Entidad estará integrada por un conjunto de registros que contendrán la información de la entidad. Existe campos opcionales como el campo de eliminación, el cual le indica al Core que un registro debe ser eliminado. Además está el campo identificador de registro nuevo, que indica que un registro es nuevo y por lo tanto se debe crear en lugar de intentar una actualización de los datos.

5.8.3. Datos de Control.

Contiene datos sueltos o campos individuales que sirven para realizar operaciones especiales con alguna transacción que lo requiera.

5.8.4. Datos de Respuesta.

Una vez ejecutado el procesamiento sobre un mensaje dado es necesario adjuntar información referente al estado del procesamiento. Para esto se utiliza esta sección, la cual incluye tres datos: código de respuesta, mensaje y stacktrace. El código de respuesta permite identificar si el procesamiento se realizó correctamente o no. En caso de una operación exitosa el código de procesamiento será OK, caso contrario en este campo se especificará el código de error obtenido. El mensaje especifica el texto que se le mostrará al usuario. En caso de error, se especifica un mensaje que le permite al usuario comprender la posible causa o reportar al administrador. Finalmente, en el campo stacktrace se adjunta una versión reducida del stacktrace obtenido en el procesamiento, únicamente en caso de error.

5.9. Diagrama de Componentes

A continuación se adjunta el diagrama de componentes, el mismo que describe los módulos en que se ha organizado la aplicación.

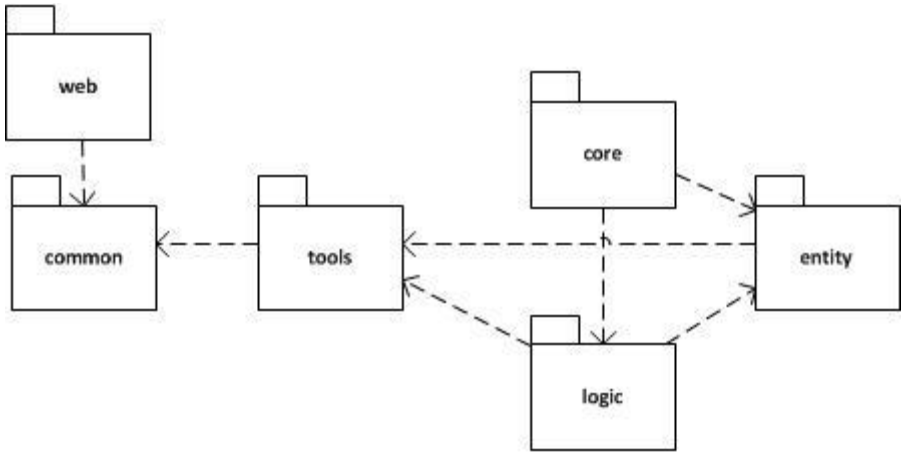


Figura 5.6 Diagrama de componentes (Módulos)

De acuerdo a la organización modular del sistema la distribución sería la siguiente:
Web: web, common
Core: core, logic, tools, common
Persistencia: entity

Cada uno de los módulos del diagrama de la Figura 5.6 a su vez, están conformados por una serie de proyectos. Esta organización se muestra en la Figura 5.7. Para una mejor visualización de esta figura referirse al Anexo 3.

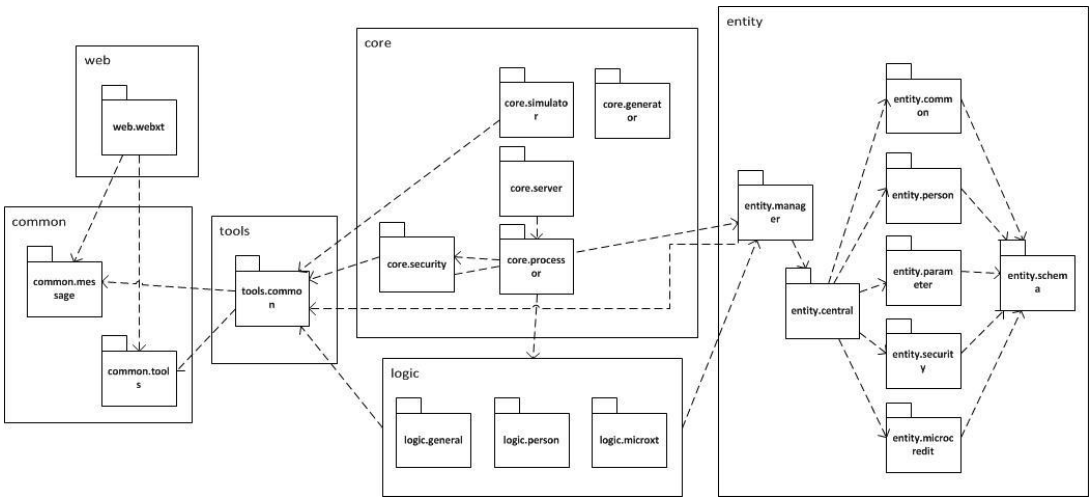


Figura 5.7 Diagrama de componentes (Proyectos)

5.10. Diagramas de clases

(Ver Anexo 4)

5.11. Diagrama ER

Para el diseño de la Base de Datos se utilizó una organización modular. Los módulos en que se dividen las tablas de la Base de datos son:

- Personas: que contiene las tablas para el manejo de la información de las personas, como datos personales, teléfonos, direcciones, etc.
- Comunes: los datos de tablas generales, como compañía, lenguaje, entre otras.
- Seguridades: contiene las tablas relacionadas con el módulo de seguridad.
- Parámetros: contiene las tablas relacionadas con el módulo de parámetros.
- Microcrédito: contiene el modelo diseñado para todo lo referente a la gestión de microcréditos, como solicitudes, clientes, zonas, etc.

Los diagramas entidad-relación, organizados por módulos se encuentran en el Anexo 5.

5.12. Diagramas de Secuencia

A continuación se muestra el diagrama de secuencia que explica de manera general como funciona el proceso elemental que realiza una transacción típica en la aplicación diseñada, desde que el usuario realiza una acción en la parte web del cliente, hasta la respuesta que recibe desde el servidor.

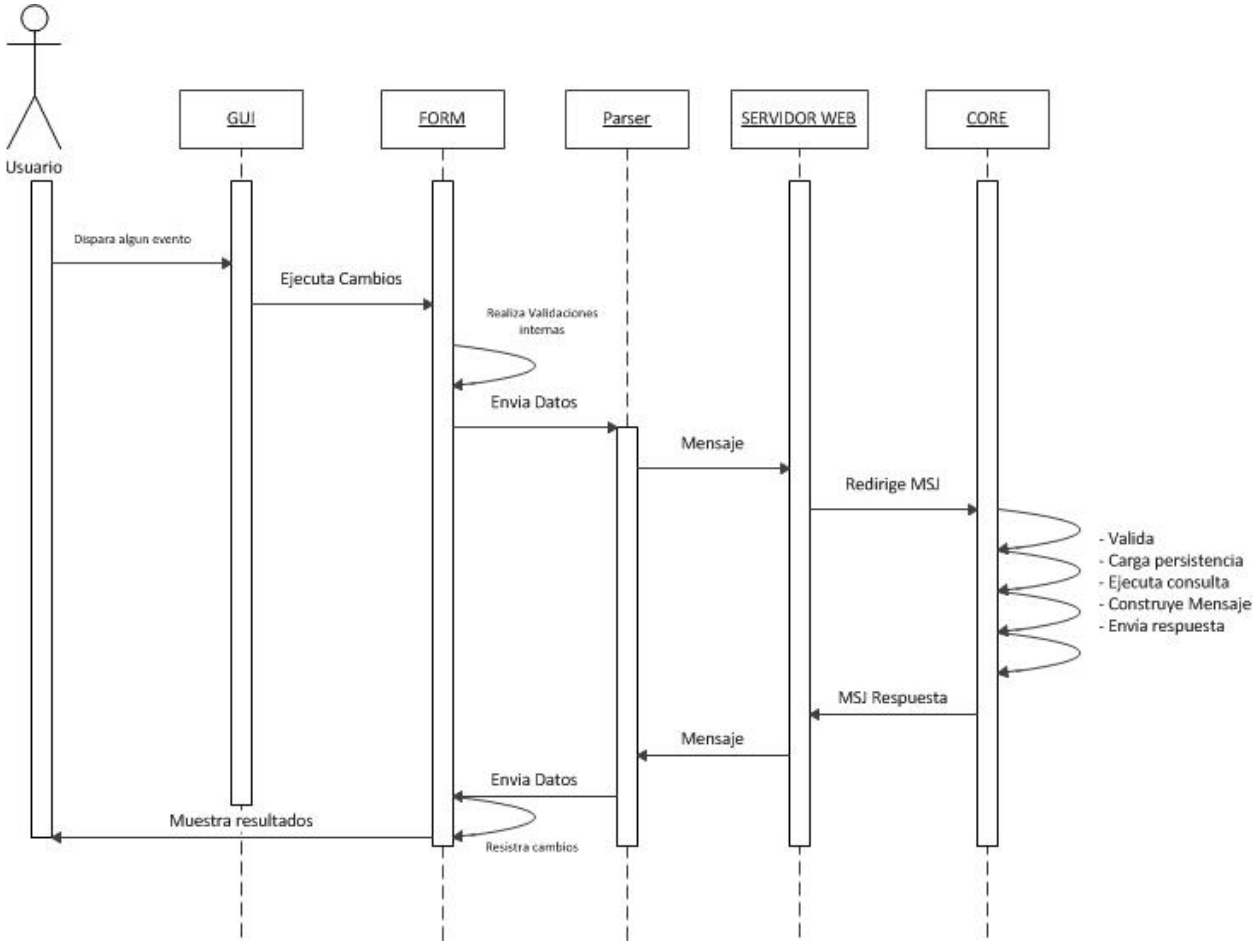


Figura 5.8 Diagrama de Secuencia General

El proceso comienza cuando el cliente accede a la aplicación y realiza una acción: esta puede ser un evento de la parte gráfica o algún mantenimiento cuando tiene que ver con la parte de procesos. Generalmente el usuario utiliza formularios para gestionar los procesos definidos, estos reciben los cambios, realizan validaciones internas y pasan modelos de datos hacia un Parser que convierte estos modelos en mensajes de tipo JSON que son enviados al servidor Web y este los redirige para que sean procesados en el CORE.

El CORE recibe los mensajes, los valida, luego convierte los datos del mensaje en entidades, utiliza la persistencia para realizar los cambios en la BD y finalmente crea un mensaje de salida con un código de respuesta indicando que el proceso ejecutado fue exitoso o no. Este mensaje se envía de retorno hasta que se hace un parseo inverso convirtiendo las datas del mensaje en elementos de un formulario, para así ser mostrados al cliente.

Para mayor detalle de los diagramas de secuencia para los procesos más importantes de la aplicación, ver el Anexo 6: Diagramas de secuencia.

5.13. Diagrama de Despliegue

A continuación se presente el diagrama de despliegue que muestra los 4 servidores empleados en este sistema:

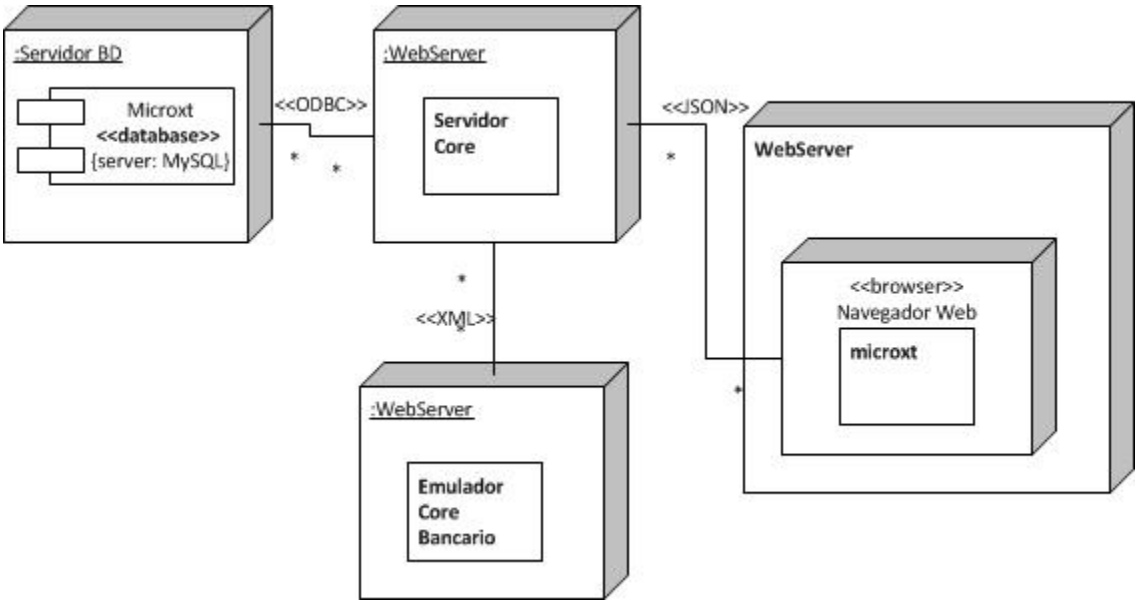


Figura 5.9 Diagrama de Despliegue

5.14. Resumen

En este capítulo se vio que la arquitectura del sistema muestra una estructura ordenada y absolutamente independiente. El medio de comunicación entre los módulos más importantes y aislados entre sí es el mensaje. También se enfatiza en la necesidad de un mensaje correctamente estructurado y extensible para facilitar y dinamizar la comunicación entre servidores. En cuanto al flujo de procesos se destacaron 3 etapas: planificación, comercialización, y finalmente la instrumentación, seguimiento y recuperación, en donde se cierra el ciclo con el représtamo. La sistematización de los procesos relacionados con dichas etapas se plasma en una organización jerárquica compuesta por Subsistema, Módulo y Proceso. En el punto 5.7.3 se plasma una tabla con los procesos implementados en el sistema de microcréditos. Finalmente se presentó algunos diagramas útiles tales como el de componentes, clases, entidad-relación, secuencia y despliegue.

Capítulo 6
Implementación

En este Capítulo se presenta un detalle de la implementación del sistema después de la etapa de diseño, se aborda primeramente la preparación e instalación del entorno de trabajo con el que se trabajó para el desarrollo del sistema, luego la parte de parametrización principalmente a nivel de Base de datos, siguiendo con la parte de la interfaz de usuario y después una explicación breve de cómo se maneja la parte del envío y recepción de mensajes entre la parte web y el servidor. Finalmente se realizan unas pruebas generales del sistema para verificar su correcto funcionamiento con todos los módulos integrados.

6.1. Estructura del Proyecto.

El sistema está configurado para ser un proyecto Maven, los módulos se acoplan perfectamente a esta estructura y con ello es posible integrar todos los proyectos tanto los de tipo java como los de tipo web. GWT también permite el uso de esta herramienta. Los módulos implementados para este proyecto se detallan a continuación:

common:
Es un módulo especial que reúne proyectos que son compartidos entre el servidor web y el servidor Core. Contiene el paquete *message*, que es el que contiene las clases que manejan la parte del mensaje XML o JSON. Y también se encuentra el paquete *tools*, donde están clases auxiliares para definir tipos de datos, conversiones, formateadores de datos y enumeraciones comunes. Estos dos proyectos son utilizados por la mayoría de los proyectos de los otros módulos.

core:
Donde se definen los paquetes que realizan el manejo de la parte del negocio y de parámetros de la aplicación, aquí se definen los servlets tanto del *core* del sistema, como del emulador del *core bancario*. Además está un paquete utilitario llamado *generator*, que sirve como herramienta para generar los scripts de parametrización de la base de datos y las entidades en base a archivos de entrada de tipo csv donde se define el modelo de las tablas que tienen que ver con el negocio (microcrédito).

entity:
En este módulo se definen los proyectos relacionados con la capa de persistencia. De esta forma, contiene un proyecto de funcionalidad que agrupa todas las clases necesarias para el manejo de la persistencia. Además están los proyectos de las clases

persistentes, que están divididas en paquetes por módulos, así hay uno para el módulo de seguridades, personas, microcrédito, comunes, parámetros, etc.

logic:
Agrupa todos los proyectos que implementan las funcionalidades del negocio o la lógica del negocio. Aquí se definen paquetes donde existen clases que para el sistema son llamadas componentes, estos son comandos utilizados por determinados procesos para ejecutar consultas o mantenimientos especiales principalmente para el módulo de microcrédito.

tools:
En este proyecto se definen herramientas que son utilizadas por los demás proyectos, posee clases con enumeraciones, conversores de tipos, definición de mensajes de estado, parámetros, además se definen interfaces para poder manejar el procesamiento de consultas y mantenimientos.

web:
Es el proyecto más grande, puesto que contiene todas las clases de la interfaz web. Desde el diseño de la interfaz por medio del modelo mvc explicado antes, la definición de los formularios (procesos), la comunicación con el servidor y la personalización de la librería de widgets provistos por GXT.

Esta estructura se define en un archivo pom.xml en la raíz del proyecto la misma que se detalla en el Anexo 7.

Se trabajó con eclipse como IDE de desarrollo. En este existen disponibles plugins para GWT y para el manejo de persistencia EclipseLink, los demás complementos como la API para el manejo de mapas, las librerías para encriptación y para pruebas *junit* se añaden desde el repositorio Maven como complementos para todos los proyectos.

No se especifica un patrón de programación determinado más que el común que se utiliza en java como notaciones, documentación, etc. Hay disponible un archivo formateador que permite extender y definir algunas propiedades extra a las disponibles por defecto en eclipse. Este archivo está incluido en las fuentes y se llama *format.xml*

El framework de GWT tiene embebido un contenedor de aplicaciones por defecto jetty, con este es posible ejecutar la aplicación en modo debug o en modo producción. Durante el desarrollo se hizo uso de este web server, y para ponerlo en producción se utiliza Apache Tomcat en su última versión 7 y de igual manera con Jetty 7.

6.2. Base de Datos y Parametrización preliminar.

Para llevar a cabo la parametrización de la BD como se mencionó en capítulos anteriores, se realizaron las configuraciones y se trabajó con las herramientas siguientes:

Base de Datos: microxt
Servidor: MySQL
Versión: 5.5.8
Notaciones y Reglas convenidas para la BD:

- No existe eliminación de registros, en lugar de ello se caducan los mismos por medio del uso de históricos.
- Los nombres de las tablas se ponen en mayúsculas y separados por el carácter "_" y en idioma Inglés.
- La fecha por defecto que se coloca en un registro insertado en el campo "EXPIRED" es 9999/12/31 00:00:00.0

Archivos de parametrización:

En el proyecto se utilizaron varios scripts de parametrización de la BD, estos están disponibles en el directorio sql de la raíz del proyecto general y están incluidos varios archivos que entre otras cosas contienen:

- *common.sql:* Llenado de tablas comunes, tipos de datos, funciones, secuenciales, etc.
- *parameter.sql:* Script que llena las tablas usadas por el módulo de Generales.
- *person.sql:* Script que llena las tablas utilizadas por el módulo de Personas.
- *security.sql:* Script que llena las tablas utilizadas por el módulo de Seguridades.
- *microcredit.sql:* Script que llena las tablas utilizadas por el módulo de Microcrédito.
- *test.sql:* Script que llena las tablas con usuarios, roles y personas iniciales, para ingreso al sistema por primera vez.

Configuración del persistence.xml:

El archivo persistence.xml es vital en el uso de JPA, ya que es el responsable de mantener la configuración para el proveedor de JPA, en la aplicación que se ejecutará debe existir en la carpeta META-INF en el directorio de mayor nivel de las clases compiladas de la aplicación. Aquí se define la conexión a la BD que tendrá esta capa de la arquitectura del sistema, del mismo modo el detalle de su configuración se encuentra especificada en el Anexo 8.

6.3. Desarrollo del módulo de Persistencia.

Tal como se indicó en capítulos anteriores la capa de persistencia fue diseñada con EclipseLink JPA, el objetivo es abstraer al modelo de negocio de todas las operaciones básicas para el manejo de tablas y relaciones de la BD y se tomaron en cuenta los siguientes aspectos en la Implementación:

Se implementaron algunas clases que se encargan de las tareas de mantenimiento y consulta de las tablas de la BD. Su funcionamiento está creado de manera tal que se automatizan los procesos de creación, actualización eliminación, manejo de históricos, control de concurrencia mediante el optimistic locking y consultas en general de las entidades del negocio.

La interacción de la capa de negocio y la de persistencia se da por medio de interfaces y utilitarios. La clase JPManager se encarga de establecer la conexión con la BD y cargar la persistencia.

Se creó una estructura jerárquica para manejar las entidades de la aplicación, dicha estructura permite el manejo y diferenciación entre las entidades normales, entidades con clave externa, aquellas con manejo de históricos, multicompañía, multilenguaje, y con manejo de optimistic locking. Además de las entidades que manejan secuenciales automáticos.

- ***Manejo de Históricos.***

Los registros de la base de datos no se eliminan, en lugar de ello se caducan. Es decir al momento de realizar un mantenimiento el registro afectado no desaparece en lugar de ello existe un campo llamado expired que cambia su valor para indicar cuál es el registro más actual. Así si un registro se caduca la fecha de este campo cambia de tener el valor por defecto (para este caso se definió una fecha que no se define vía calendario 9999-12-31 00:00:00.0), a tener la fecha y hora de modificación. Esto puede servir para motivos de Auditorias posteriores o simplemente para llevar registro de todos los cambios en la BD.

- ***Multicompañía.***

Algunas tablas comparten la característica de tener un campo llamado company, con este se define la posibilidad de tener registros y tablas según la compañía. Para esta Aplicación se trabaja con el nombre MXT.

- ***Multilenguaje.***

Algunas tablas comparten la característica de tener un campo llamado language, con este se define la posibilidad de tener registros y tablas con determinado lenguaje. Para esta Aplicación y dado que va orientado a zonas dentro del Ecuador, el lenguaje se define por ES. (Español).

- ***Optimistic Locking***

Las transacciones que lanzan más de una vez una misma consulta, tienen que implementar un nivel de aislamiento mayor, por lo tanto, esta estrategia es la más recomendada por JPA, ya que maximiza la eficiencia, a menor nivel de aislamiento menos bloqueo en la Base de Datos. Además por ser la más adecuada para la mayor parte de las transacciones. Para implementar este tipo de estrategias en nuestras aplicaciones se debe añadir un atributo o propiedad de nombre versión en las entidades modificables.

El paquete manager trabaja con la capa de persistencia y parsing entre los datos del mensaje a entidades en la clase JPManager.

Se trabaja con entidades que están separadas del contexto de la BD, de manera que cualquier cambio o alteración que se pueda ejecutar sobre las entidades en la etapa de procesamiento no se verán reflejadas en la BD a menos q se lo indique explícitamente.

6.4. Desarrollo del módulo Core.

El módulo Core es una de las partes más importantes del sistema. Aquí se definen los dos servlets disponibles que se encargan de manejar los procesos de negocio y el simulador respectivamente:

Core Server.

Donde se define el servlet Core.java que es un HttpServlet y se encarga de recibir el mensaje enviado desde la parte web cuyo formato es JSON, parsea su contenido, y utiliza la clase CoreProcessor para manejar las persistencias convirtiendo ítems a Entidades por medio del JPManager. Además la clase PersistenceListener.java que es la encargada de levantar las persistencias definidas en el paquete entity al iniciar las operaciones. El archivo de configuración del Servlet web.xml consta en el Anexo 9.

Core Simulator.

Donde se define el servlet Simulator.java que igualmente es un HttpServlet y se encarga de recibir el mensaje enviado desde Core Server cuyo formato es XML, parsea su contenido, y utiliza la clase SimulatorProcessor para generar la tabla de pagos de las solicitudes enviadas por el asesor al momento de realizar la instrumentación. El archivo de configuración del Servlet web.xml consta en el Anexo 10.

Además de los servlets que ejecutan y manejan procesos de negocio, se definen las clases que trabajan con el mensaje para procesarlo, convertirlo en entidades y así poder trabajar con las persistencias, armar las consultas SQL y ejecutarlas, finalmente devolver el mensaje de respuesta para visualizarla al cliente por medio del módulo WEB. Las clases mencionadas se encuentran en el paquete processor y son las siguientes:

- *CoreProcessor.java*
- *GeneralMaintenance.java*
- *GeneralQuery.java*

En el paquete security se definen todos los procesos de control tanto de inicio y fin de sesión, validaciones y asignaciones de terminales, roles y perfiles.

En el módulo CORE se procesan los códigos de error generados ya sea en la Base de Datos o a nivel de aplicación. Cuando salta un error a nivel de Base de datos, JPA muestra un stacktrace que notifica el tipo de error y cuál fue su causa, de igual forma lo hace el framework de Java, con los códigos de error generados se definen textos personalizados que sirven para enviarlos de vuelta en el mensaje de respuesta y así se visualizan en el front-end al cliente de manera que pueda identificar fácilmente en caso de que algo haya fallado.

6.5. Desarrollo del módulo Web.

La estructura del proyecto web está definida como se muestra y explica brevemente a continuación:

PAQUETE	DESCRIPCIÓN
mobile.web.webxt.client	Configuración del entorno y carga de los componentes de la interfaz.
mobile.web.webxt.components	Personalización de componentes de la GUI
mobile.web.webxt.data	Comunicación con el CORE, proxies, stores y configuraciones en general.
mobile.web.webxt.data.form	Diseño de formularios
mobile.web.webxt.devform	Procesos del negocio. (Formularios)
mobile.web.webxt.form	Widgets utilizados por los formularios
mobile.web.webxt.form.validation	Validaciones para los widgets utilizados por los formularios
mobile.web.webxt.form.widgets	Widgets personalizados para los componentes de la GUI.
mobile.web.webxt.mvc	Definición del modelo mvc
mobile.web.webxt.parser	Paseador de mensajes XML y JSON
mobile.web.webxt.resources	Imágenes, Iconos, etc. Utilizados para la GUI
mobile.web.webxt.treemenu	Definición y Configuración del árbol de procesos
mobile.web.webxt.util	Utilitarios como convertidores, formateadores, tipos de datos manejados en la GUI.
mobile.web.webxt.windows	Ventanas de notificación para el usuario

Tabla 6.1 Estructura del proyecto web

GWT organiza un proyecto web por módulos, estos son definidos en ficheros XML cuya extensión de archivo es .gwt.xml. Aquí se definen las librerías a ser referenciadas por el proyecto, por ejemplo las de comunicación con servlets, API de mapas, encriptación, etc. Además se define el punto de entrada que es el archivo principal que se ejecuta al inicio y finalmente se configuran algunos aspectos de optimización y otros de idioma o internacionalización. La definición del archivo *Mobile.gwt.xml*, está en el Anexo 11.

6.5.1. Página Principal.

Al acceder a la Aplicación, la primera pantalla que se visualiza es la carga de la aplicación seguida de la ventana de inicio de sesión, donde se pide el usuario y la contraseña. Si el acceso es correcto se visualiza la ventana de roles, si el usuario tiene más de un rol definido, da la opción para escoger con cual acceder. Según el rol escogido se carga la página principal de la aplicación, en caso de ser Administrador, este tiene acceso a todos los procesos del negocio. Si es un Asesor el que ingresa, este solo tiene acceso a los procesos definidos para la comercialización, venta y recuperación en la gestión del microcrédito.

6.5.2. Módulos y Procesos.

Tal como se especificó en el diseño del sistema, la página principal cuenta con 3 partes principales: El árbol de procesos, La ventana de formularios y el área de notificación o barra de estado. Se implementaron los 4 módulos principales del negocio: *GENERALES, SEGURIDADES, PERSONAS Y MICROCRÉDITO*.

Los procesos son tipos de formularios donde el usuario ingresa o consulta información; gracias a la funcionalidad del modelo mvc, el usuario selecciona un proceso del árbol o de los accesos directos de la página principal y en la ventana de formularios se cargan estos en un TabPanel. (Figura 6.1)

Figura 6.1 GUI de la Aplicación

6.6. Comunicación con el servidor.

Tal como se planteó en la etapa de diseño de la aplicación, la comunicación con el servidor es a través de mensajes de tipo JSON. El proceso comienza al ejecutar el servlet del CORE que como se mencionó en capítulos anteriores, es el responsable de levantar la persistencia. Luego el usuario accede al sistema y realiza alguna acción. El log del CORE muestra cómo se envía y se recibe el mensaje, a continuación se muestra una secuencia como ejemplo de cómo se realiza este proceso.

1. *Ejecución del Core Server y Levantamiento de la persistencia (Figura 6.2).*

```
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building MOBILE CORE SERVER 2.0
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] >>> jetty-maven-plugin:7.0.0.pre5:run-exploded (default-cli) @ mobile.cor
.server >>>
[INFO]
[INFO] --- maven-resources-plugin:2.5:resources (default-resources) @ mobile.cor
.server ---
[debug] execute contextualize
[INFO] Using 'UTF-8' encoding to copy filtered resources.
```

...

```
[INFO] Configuring Jetty for project: MOBILE CORE SERVER
2012-04-02 11:46:32.454::INFO:  Logging to STDERR via org.mortbay.log.StdErrLog
[INFO] Context path = /mobile
[INFO] Tmp directory =  determined at runtime
[INFO] Web defaults = org/mortbay/jetty/webapp/webdefault.xml
[INFO] Web overrides =  none
[INFO] Starting jetty 7.0.0.pre5 ...
2012-04-02 11:46:33.951::INFO:  jetty-7.0.0.pre5
INFO  [main] (PersistenceListener.java:26) - mobile.core.server.PersistenceListe
ner constructor
```

...

```
[EL Config]: The alias name for the entity class [class mobile.entity.parameter.
ProvinceId] is being defaulted to: ProvinceId.
[EL Config]: The alias name for the entity class [class mobile.entity.security.M
odule] is being defaulted to: Module.
[EL Config]: The alias name for the entity class [class mobile.entity.microcredi
t.ZoneAsessor] is being defaulted to: ZoneAsessor.
[EL Config]: The alias name for the entity class [class mobile.entity.common.Dat
abaseType] is being defaulted to: DatabaseType.
[EL Config]: The alias name for the entity class [class mobile.entity.person.Per
sonType] is being defaulted to: PersonType.
[EL Config]: The alias name for the entity class [class mobile.entity.security.P
rocess] is being defaulted to: Process.
INFO  [main] (JPManagerFactory.java:36) - Load time for persistence: 00:02.886
INFO  [main] (PersistenceListener.java:39) - Initialize global parameters
2012-04-02 11:46:58.842::INFO:  Started SelectChannelConnector@0.0.0.0:9090
[INFO] Started Jetty Server
```

Figura 6.2 Secuencia de ejecución del CORE server

2. *El usuario realiza una acción en la aplicación, que puede ser una consulta, mantenimiento o ingreso nuevo y la respuesta en el servidor se manifiesta por medio del mensaje de entrada. En la* Figura 6.3 *se coloca una captura del log del servidor core en la parte donde muestra el mensaje de entrada para el proceso C101, con procesamiento de tipo Consulta (QRY):*

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
  <data id="request">
    <user>ADMIN</user>
    <host>LOCALHOST</host>
    <channel>PC</channel>
    <session>FCD708B8E428C1FB7B04EAF180EBF3232012040202022093 6</session>
    <profile>ADM</profile>
    <process>C101</process>
    <type>QRY</type>
  </data>
  <data id="control">
    <_generatedId/>
  </data>
  <data id="Person">
    <_alias>per</_alias>
    <_offset_page>0</_offset_page>
    <_filters>pk_personId:=:1</_filters>
    <_limit_page>1</_limit_page>
    <_qry_fields>name;lastName;secondLastName;dateOfBirth;genderTypeId;GenderTyp
e.name;civilStatusId;CivilStatus.name;professionTypeId;ProfessionType.name;ident
ificationTypeId;IdentificationType.name;identificationNumber;countryId;Country.n
ame:pk_countryId=per=countryId;provinceId;Province.name:pk_countryId=per=country
Id^pk_provinceId=per=provinceId;cityId;City.name:pk_countryId=per=countryId^pk_p
rovinceId=per=provinceId^pk_cityId=per=cityId;districtId;District.name:pk_countr
yId=per=countryId^pk_provinceId=per=provinceId^pk_cityId=per=cityId^pk_districtI
d=per=districtId</_qry_fields>
  </data>
</message>
```

Figura 6.3 Ejemplo mensaje de entrada

3. *Luego el mensaje es procesado y devuelve el mensaje de respuesta (Figura 6.4):*

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
  <data id="request">
    <user>ADMIN</user>
    <host>LOCALHOST</host>
    <channel>PC</channel>
    <session>FCD708B8E428C1FB7B04EAF180EBF3232012040202020220936</session>
    <profile>ADM</profile>
    <process>C101</process>
    <type>QRY</type>
  </data>
  <data id="control">
    <_generatedId/>
  </data>
  <data id="Person">
    <_alias>per</_alias>
    <_offset_page>0</_offset_page>
    <_filters>pk_personId:=:1</_filters>
    <_limit_page>1</_limit_page>
    <_qry_fields>name;lastName;secondLastName;dateOfBirth;genderTypeId;GenderTyp
e.name;civilStatusId;CivilStatus.name;professionTypeId;ProfessionType.name;ident
ificationTypeId;IdentificationType.name;identificationNumber;countryId;Country.n
ame:pk_countryId=per=countryId;provinceId;Province.name:pk_countryId=per=country
Id^pk_provinceId=per=provinceId;cityId;City.name:pk_countryId=per=countryId^pk_p
rovinceId=per=provinceId^pk_cityId=per=cityId;districtId;District.name:pk_countr
yId=per=countryId^pk_provinceId=per=provinceId^pk_cityId=per=cityId^pk_districtI
d=per=districtId</_qry_fields>
    <_total_page>1</_total_page>
    <item number="1">
      <name>RONALD MARCELO</name>
      <lastName>GUAL┴N</lastName>
      <secondLastName>SAAVEDRA</secondLastName>
      <dateOfBirth>(Date)1988-09-28</dateOfBirth>
      <genderTypeId>M</genderTypeId>
      <GenderType_name>MASCULIÑO</GenderType_name>
      <civilStatusId>SOL</civilStatusId>
      <CivilStatus_name>SOLTERO</CivilStatus_name>
      <professionTypeId>101</professionTypeId>
      <ProfessionType_name>QUIMICO</ProfessionType_name>
      <identificationTypeId>CED</identificationTypeId>
      <IdentificationType_name>CEDULA DE IDENTIDAD</IdentificationType_name>
      <identificationNumber>1400658611</identificationNumber>
      <countryId>EC</countryId>
      <Country_name>ECUADOR</Country_name>
      <provinceId>AZ</provinceId>
      <Province_name>AZUAY</Province_name>
      <cityId>CU</cityId>
      <City_name>CUENCA</City_name>
      <districtId>RIC</districtId>
      <District_name>RICAURTE</District_name>
    </item>
  </data>
  <data id="response">
    <code>OK</code>
    <message>PROCESO EXITOSO</message>
  </data>
</message>
```

Figura 6.4 Ejemplo mensaje de salida

Como se ve en la secuencia de imágenes mostrada antes, esa es la forma de comunicación cliente-servidor. Una parte importante a destacar es la estructura del mensaje, que fue explicado en el capítulo 6 (Ver sección 5.8):

6.7. Pruebas generales.

A continuación se muestran algunas de las pruebas realizadas que demuestran las funcionalidades y la capacidad para manejar la información, eventos y la comunicación con el servidor, por medio de algunas de las transacciones más importantes tanto de la plataforma como de la interfaz de usuario.

- ✓ ***Carga de Accesos directos para procesos más Usados:***

Esta funcionalidad permite acceder rápidamente a los procesos más utilizados ya sea por el Asesor o el Coordinador de microcrédito, fue diseñada para trabajar de manera más versátil con dispositivos tipo Tablet, dado que se utilizó un modelo mvc, tanto el árbol de procesos como la pantalla de accesos y los tabs, están sincronizados entre sí. A continuación, en la Figura 6.5 se muestra una captura de la interfaz principal de la aplicación. El componente con los accesos directos está en el tab de título Principal, que es un tab fijo de la ventana.

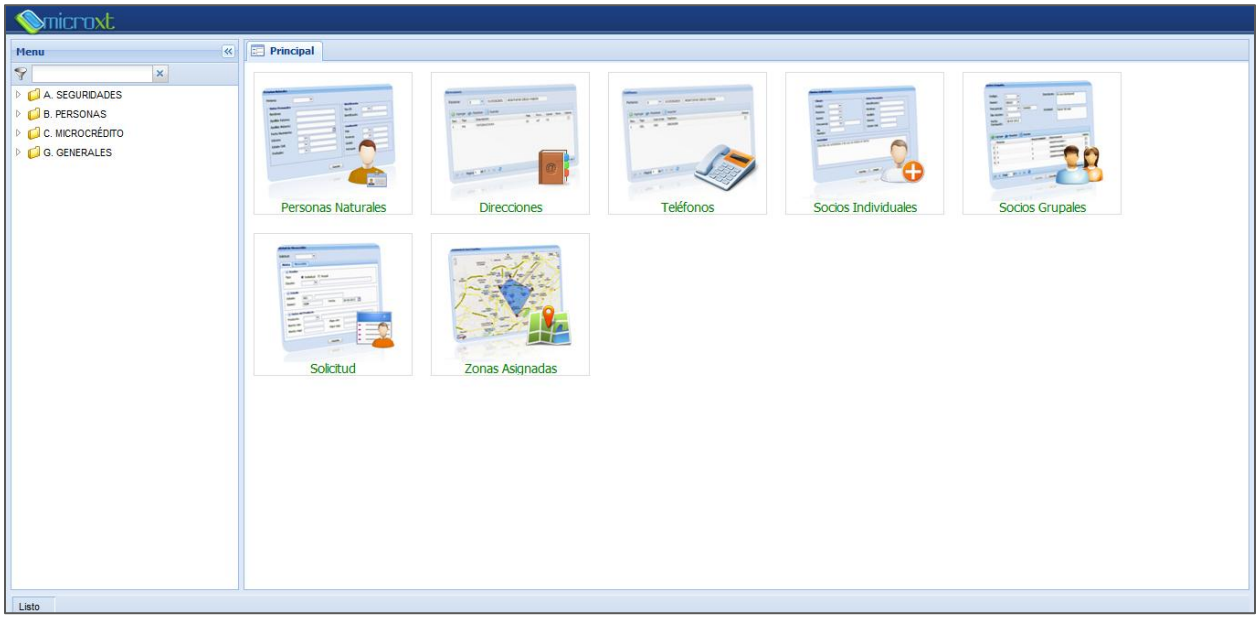


Figura 6.5 Pantalla principal de la Aplicación

- ✓ ***Formularios tipo grilla para mantenimientos:***

El formulario más común en el flujo de procesos es de tipo grilla, por medio de este se pueden ejecutar los tres eventos básicos contra una o varias tablas de la base de datos: consulta, mantenimiento y cuando se caduca un registro. En la Figura 6.6 se muestra un formulario tipo grilla de ejemplo.

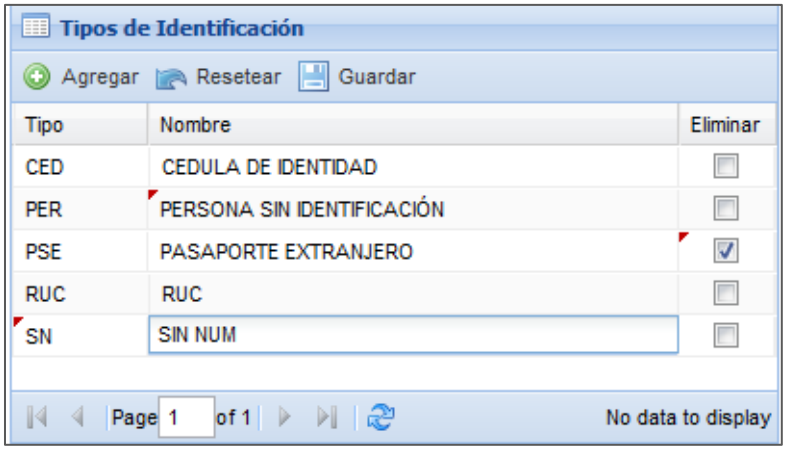


Figura 6.6 Grilla de mantenimientos

- ✓ ***Filtros, paginación y validaciones en grillas:***

Los formularios tipo grilla tienen varias funcionalidades, entre ellas las validaciones de tipos de datos o validaciones personalizadas, la paginación y el listado de registros en base a filtros de distintos tipos. La función de validación se ilustra en la Figura 6.7, mientras que los filtros se muestran en la Figura 6.8.

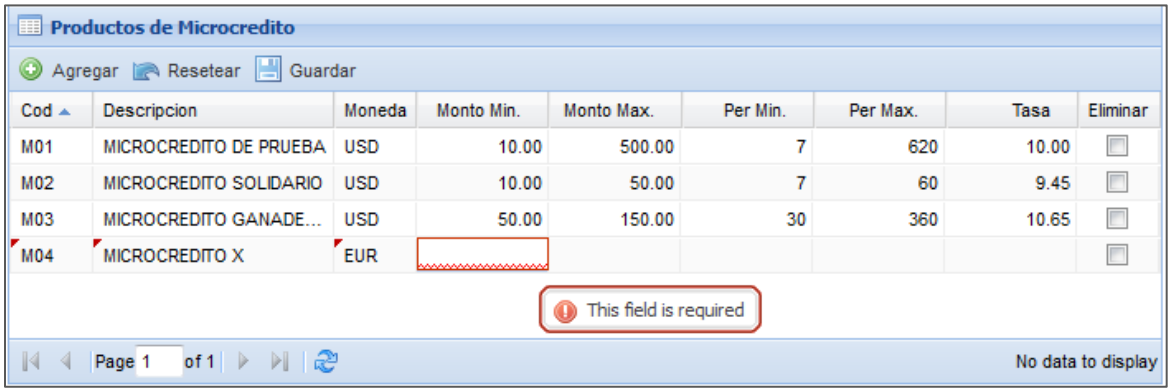


Figura 6.7 Validaciones en grillas

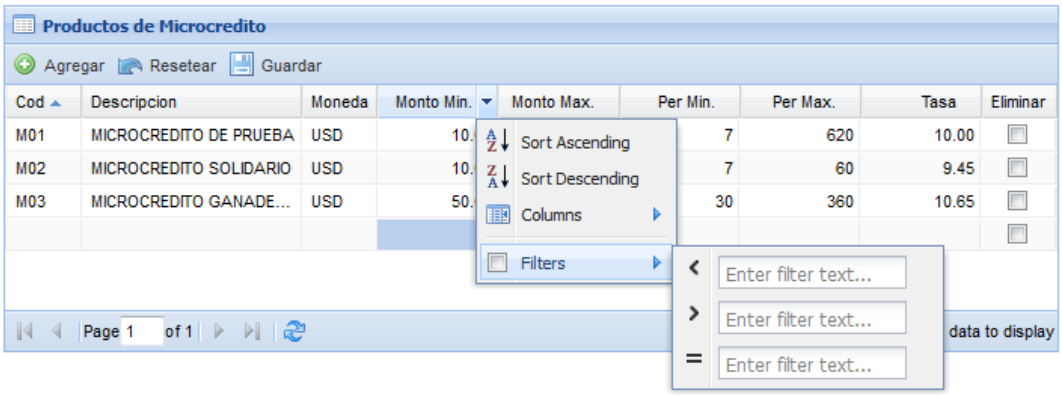


Figura 6.8 Filtros en Grillas

- ✓ ***Formularios normales:***

El otro tipo de formulario utilizado en el flujo de procesos es de tipo form, por medio de este se hace ingresos, consultas y actualizaciones generalmente de registros individuales de una determinada tabla. De igual manera, tiene funcionalidades como validaciones automáticas, listas de valores simples y con paginación, formateadores, etc. En la Figura 6.9 se muestra el formulario de Solicitud de Microcrédito.

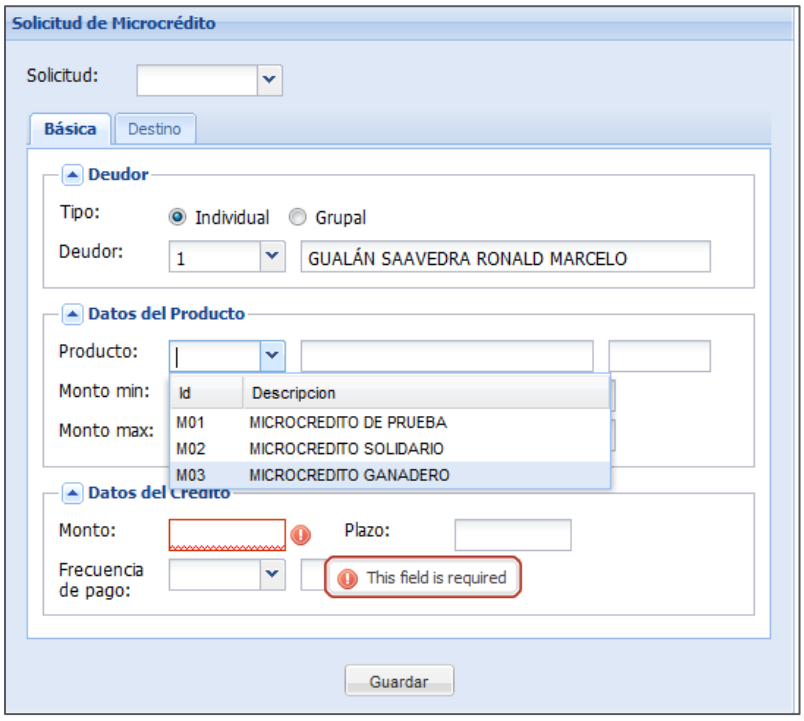


Figura 6.9 Ejemplo de formulario normal

- ✓ ***Formularios maestro-detalle:***

Los formularios tipo maestro-detalle tienen un nivel de complejidad más elevado, porque utilizan 2 o más orígenes de datos (2 o más tablas). La Figura 6.10 señala un formulario de este tipo que extrae los datos de la tabla de clientes grupales y de la tabla de miembros del grupo.

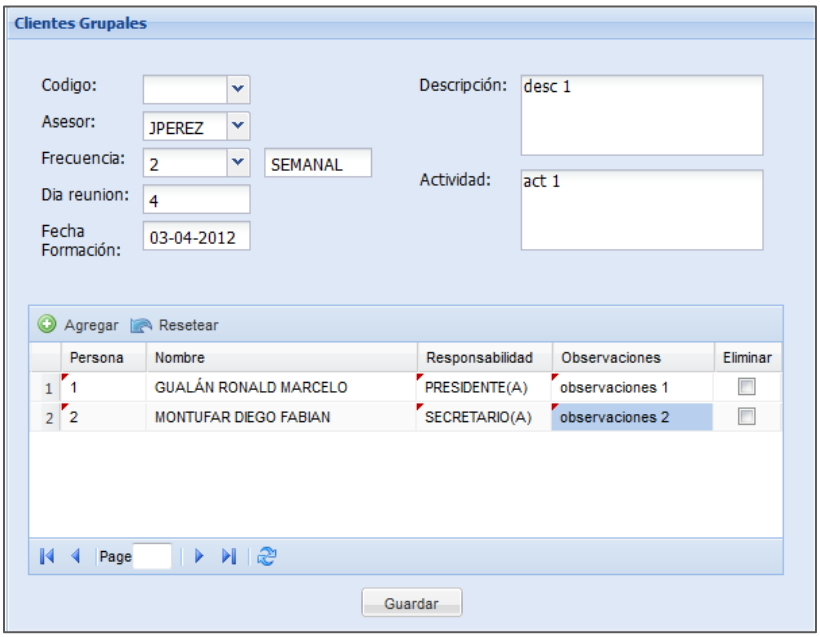
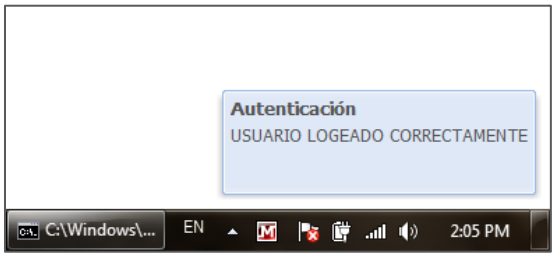


Figura 6.10 Ejemplo de formulario maestro-detalle

- ✓ ***Notificaciones al usuario:***

Las notificaciones al usuario se pueden presentar de tres maneras principalmente: Mensajes flotantes que aparecen en la esquina inferior derecha (Figura 6.11, primera captura), mensajes en la barra de estatus (Figura 6.11, segunda captura) o mensajes tipo ventana (Figura 6.11, tercera captura).



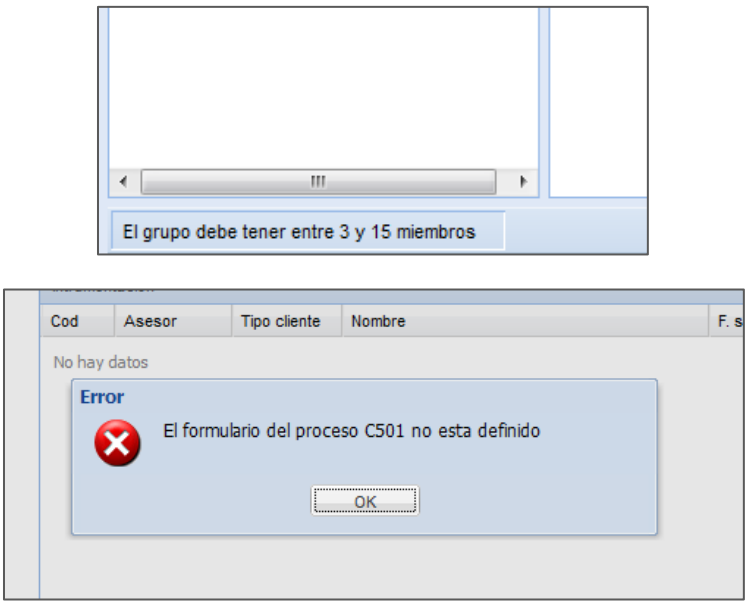


Figura 6.11 Notificaciones al Usuario

6.8. Resumen

Al finalizar el capítulo que trata sobre la implementación del sistema, se puede destacar algunos aspectos, como la organización de los módulos a implementar. En donde se presentó una organización basada en seis módulos: common, core, entity, logic, tools y web. En lo relacionado con el desarrollo del módulo de persistencia se mencionó la posibilidad de manejar las tablas con características como seguimiento de movimientos históricos, tener datos multicompañía y multilenguaje, y poder manejar concurrencia mediante el mecanismo conocido como Optimistic Locking. Por otro lado, se abordó la existencia de dos servidores de procesamiento, el primero el servidor principal (servidor core) y el segundo un servidor de simulación, que interactúa con el primero. Finalmente se visualiza algunos aspectos de la puesta en escena de la aplicación mediante el uso de capturas de pantalla.

Capítulo 7
Resultados

7.1. Conclusiones

Durante el desarrollo del proyecto se obtuvo una gran cantidad de experiencia en todas las áreas aplicadas y entre otras cosas se pudo constatar la utilidad, la eficacia, las ventajas y desventajas de las herramientas de software que se emplearon. A continuación se describen algunas de las conclusiones a las que se llegó una vez finalizada la implementación de la aplicación para la gestión de microcréditos:

- ***Uso del Modelo Ágil.***
  Varios métodos de desarrollo de software tratan de reducir el tiempo entre la fase de análisis y la exigencia en la fase de desarrollo para construir algunas partes del sistema en menos tiempo, posiblemente con un método iterativo para avanzar a la aplicación final. En esta tesis se siguió la metodología AUP, que fue de gran beneficio dado a que sigue un proceso iterativo que libera prototipos, lo cual ayudó en todas las etapas de la construcción del sistema, desde el diseño de la arquitectura hasta la implementación, permitiendo realizar cambios importantes que contribuyeron al desarrollo de una aplicación más estable y mejor estructurada. GWT se adapta perfectamente a esas metodologías, ya que puede ofrecer un desarrollo iterativo, prototipado, rápido y ofrece pruebas automatizadas.

- ***Configuración del Entorno:***
  Como herramientas utilizadas para trabajo en grupo se utilizaron las provistas por Apache Maven y Subversion. La primera sirvió para organizar la aplicación en módulos y proyectos. Gracias a Maven es posible compilar, empaquetar, generar documentación, pasar los test y preparar los entregables de los proyectos fácilmente. Esto representó una excelente herramienta para la gestión de este proyecto. GWT es totalmente compatible con Maven y su configuración es relativamente sencilla. Subversion también fue una herramienta muy útil que permitió manejar el proyecto de manera colaborativa al tenerlo subido en un repositorio accesible por los desarrolladores.

- ***Características de GWT***
  GWT es una herramienta muy poderosa y en términos de estabilidad y eficacia sobresale ante otras alternativas, sin embargo durante el desarrollo de esta aplicación se pudieron detectar algunos puntos en contra. Gracias a la extensión Ext GWT (Sencha), fue posible disponer de una gama de componentes gráficos

muy amplia y en general el manejo de la GUI no representó ningún inconveniente, excepto en la parte de la comunicación con el servidor. El problema surgió en principio, al intentar usar una comunicación cliente-servidor por medio de RPC, debido a que la capa de persistencia es incompatible con este método. Sin embargo, este mecanismo no fue utilizado no solamente por esa razón, sino también porque el enfoque inicial planteado para la comunicación cliente-servidor era por medio de mensajes XML o JSON.

- ***SOP (Same Origin Policy, Política del mismo origen)***
  Durante el desarrollo de los mecanismos de comunicación por medio de mensajes entre el front-end (web) y el back-end (core) surgió un problema debido a una política de restricción llamada SOP. La misma que impide a una aplicación web crear requerimientos Ajax hacia otras direcciones IP diferentes a la del servidor web. La restricción persiste incluso cuando las peticiones son hacia la misma IP pero con puerto diferente. Esto se debe a que SOP es una política por medio de la cual los navegadores restringen la modificación de páginas/recursos -generalmente con Javascript- que usan o intentan acceder a contenidos obtenidos de otro lugar diferente al "origen". Esta política se puede saltar valiéndose del concepto de CORS (Cross Origin Resource Sharing) o directamente utilizando una librería de XDM (Cross Direct Messaging). Para esta aplicación se utilizó la primera solución, utilizando un Proxy en el servidor web de webxt que redirige todos los mensajes XML hacia el servidor core, eliminando así la restricción SOP y permitiendo utilizar la comunicación cliente-servidor con mensajes XML o JSON.

- ***GWT multiplataforma.***
  Según la página de desarrollo de GWT, indica que es compatible y funciona igual con los navegadores: Firefox, Internet Explorer 6 y 7, Safari, Opera 9.0 y obviamente con el navegador Chrome. Para la aplicación desarrollada se hicieron pruebas en todos los navegadores mencionados y se encontraron algunas diferencias que si bien no son muy importantes, pero representan una problemática considerable en cuanto a términos de interfaz de usuario y a desempeño de la aplicación desarrollada. Chrome es perfectamente compatible, permite tanto el modo alojado como el modo de producción. Algo que no es posible con versiones más recientes de Firefox e IE. En este último funcionan ambos modos de ejecución siempre y cuando la última versión (9.0) haya sido configurada para compatibilidad con las versiones 6 o 7 en modo desarrollo. Aquí los controles, por ser modo compatibilidad, no se muestran bien en un 100%.

- ***Manejo de Mensajes XML-JSON***
  Como se ha mencionado a lo largo de este documento, la aplicación maneja mensajes de tipo XML o JSON para la comunicación entre el cliente y el

servidor. La ventaja de este tipo de archivos es su portabilidad y que son relativamente livianos, fáciles de entender y principalmente son estándares ampliamente utilizados. Es por ello que, al haber sido diseñados de manera personalizable, es posible conectar la aplicación a servidores externos. Para el caso de la aplicación de microcrédito se utiliza un emulador de Core Bancario. La utilización de estos mensajes se puede hacer indistintamente. Aunque XML y JSON tienen algunas diferencias estructurales que pueden crear incompatibilidades, es posible manejarlos sin mayor problema gracias a la flexibilidad que ambos ofrecen.

- ***Tecnologías recientes con bugs.***
  Uno de los objetivos de la presente tesis fue la de aprender y utilizar algunas de las tecnologías más recientes y de mayor auge tanto para el trabajo con el front-end como para el manejo de la capa de persistencia para la base de datos. En el caso del front-end se decidió trabajar con GWT por medio de una Extensión llamada Ext GWT. Mientras que para el manejo de la persistencia con JPA se acordó usar EclipseLink. Ambas son excelentes alternativas, con grandes cualidades y un futuro prometedor. Sin embargo, al tratarse de plataformas relativamente nuevas poseen falencias relacionadas con bugs que no han sido corregidos o desarrollados completamente. Esto hace que existan problemas durante el desarrollo, y que haya que buscar alternativas para el manejo de dichos errores. La buena noticia al respecto, es que las organizaciones que se encuentran detrás de estas tecnologías trabajan continuamente para mejorar estos problemas y así es como van sacando nuevas versiones cada vez con mejores características y menos errores. De allí que, al utilizar tecnologías sujetas a constante evolución, es recomendable ir actualizando las aplicaciones, de manera que se puedan ir beneficiando de las mejoras insertadas en las tecnologías empleadas.

- ***Sobre los resultados.***
  Finalmente al respecto de los resultados de esta tesis se puede decir que se logró cumplir a cabalidad con los objetivos planteados. Uno de los más importantes era el poder elaborar una plataforma bien organizada que tuviera todas las características propias de un sistema escalable y fácil de mantener sobre el que se puede desarrollar cualquier aplicación sin mayor dificultad y con un gran alcance en cuanto a dispositivos. Y en efecto, gracias a las características planteadas y a las tecnologías empleadas fue posible emplear la arquitectura para el desarrollo de una aplicación de microcréditos con una dificultad media-alta. Uno de los indicadores más importantes, es el hecho de haber podido cumplir con las expectativas de los interesados en esta tesis, obteniendo siempre buenos comentarios al respecto.

7.2. Trabajos Futuros

La Aplicación Web para la gestión de Microcréditos, es un sistema cuya arquitectura fue diseñada para ser escalable y parametrizable, con el objetivo de permitir realizar mejoras permanentes en todos los ámbitos. Desde la estructura de la BD, pasando por el diseño Web, Interfaz de usuario, definición de procesos hasta la conexión del sistema con otros servicios web que utilicen estándares utilizados por la misma. En general el sistema se considera altamente compatible y no orientado a un solo objetivo debido a su estructura genérica. Para trabajos futuros se recomienda lo siguiente:

- ***Organización y trabajo en grupo.***
  El sistema fue estructurado como proyecto Maven-Subversion, lo recomendable sería seguir trabajando de la misma manera, utilizar repositorios remotos o locales y manejar control de cambios para llevar el desarrollo de la manera más organizada posible. Esto produce resultados beneficiosos y permite un crecimiento y mantenimiento más sencillo del sistema.

- ***Conexión con otros sistemas.***
  Dado que se desarrolló una aplicación orientada a la banca y las finanzas y se utilizó un emulador de CORE bancario para simular la conexión con un sistema externo, se recomienda poner a prueba la capacidad de conexión mediante la forma de comunicación de mensajes XML con sistemas externos reales.

- ***Generalizar Componentes.***
  El presente proyecto aprovecha los componentes y widgets de la librería GWT y ExtGWT de manera que es posible tener otros componentes, pero más personalizados y en algunos casos más óptimos en desempeño. Esto se ha logrado gracias a la abstracción de los elementos provistos por la plataforma de GWT y por la construcción genérica de la parte web del sistema. Se recomienda seguir este estándar con el fin de reutilizar el desarrollo en otro tipo de sistemas, evitando la pérdida de tiempo y funcionalidad de aplicaciones con código redundante y poco reutilizable.

- ***Extender funcionalidad para dispositivos móviles.***
  Una de las mayores posibilidades en cuanto a la mejora del presente sistema podría estar centrada en un mejor aprovechamiento de las capacidades de los dispositivos móviles. Así por ejemplo, una mejora significativa para el sistema, podría ser el uso de la cámara de los dispositivos móviles. Esto podría permitir que se pueda tomar fotos de los documentos de respaldo, en el caso de la aplicación de microcrédito, por ejemplo. Las posibilidades son muchas en este campo.

Referencias

[1] T. Jansson, *Financiar las Microfinanzas*. Washington, D.C: Banco Interamericano de Desarrollo, 2003.

[2] C. Ortega, *Finanzas populares y Migración: tejiendo la red para el desarrollo local*. Quito: Ediciones FEPP, 2006.

[3] PortalMicrofinanzas. (2009) PortalMicrofinanzas. [Online]. http://www.portalmicrofinanzas.org/p/site/s/template.rc/1.26.11201/

[4] Scott W. Ambler, *A Manager's Introduction to the Rational Unified Process (RUP)*., 2009. [Online]. http://www.ambysoft.com/unifiedprocess/rupIntroduction.html

[5] G. Vasconez, *Modelo de manual de politicas y procedimientos de microcredito para Cooperativas de Ahorro y Credito*. Quito: DGRV, 2005.

[6] G. Vásconez, *Manual Metodológico de Microcrédito individual para Cooperativas de Ahorro y Crédito (CACS)*. Quito: DGRV, 2005.

[7] Grameen Foundation USA. (2003, Enero) Grameen Foundation USA. [Online]. http://www.gfusa.org

[8] Flurry. (2011, Octubre) Flurry. [Online]. http://www.flurry.com

[9] Adobe Systems Inc. (2012) Phonegap. [Online]. http://phonegap.com/

[10] Web Directions. (2011) Web Directions. [Online]. http://www.webdirections.org/

[11] Mike Keith and Merrick Schincariol, *Pro JPA 2. Mastering the Java Persistence API*.: APRESS, 2009.

[12] The Eclipse Foundation. (2011) EclipseLink Documentation Center. [Online]. http://wiki.eclipse.org/EclipseLink/UserGuide

[13] Harold Elliotte and Scott Means, *XML in a Nutshell. A desktop quick reference*.: O'REILLY, 2004.

[14] JSON ORG. (2011) Introducing JSON. [Online]. http://json.org/

[15] Google INC. (2011) Google Web Toolkit. [Online]. https://developers.google.com/web-toolkit/

[16] Daniel Vaughan, *Ext GWT 2.0. Beginner's Guide*.: Packt Publishing, 2010.

[17] NASOFT. (2008) NASOFT. [Online]. http://www.nasoft.com/site/Home/AcercadeNasoft/Qui%C3%A9nessomos/tabid/83/Default.aspx

[18] Fundación Wikimedia, Inc. (2012) Wikipedia. [Online]. www.wikipedia.org

[19] G. Vasconez, *Modelo de manual de politicas y procedimientos de microcredito para Cooperativas de Ahorro y Credito*. Quito: DGRV, 2005.

[20] T. Jansson, *Financiar las Microfinanzas*. Washington, D.C: Banco Interamericano de Desarrollo, 2003.

[21] PortalMicrofinanzas. (2009) portalmicrofinanzas. [Online].

http://www.portalmicrofinanzas.org/p/site/s/template.rc/1.26.11201/

[22] NASOFT. (2008) NASOFT. [Online]. http://www.nasoft.com/site/Home/AcercadeNasoft/Qui%C3%A9nessomos/tabid/83/Default.aspx

[23] G. Vásconez, *Manual Metodológico de Microcrédito individual para Cooperativas de Ahorro y Crédito (CACS)*. Quito: DGRV, 2005.

ANEXOS

ANEXO 1. Detalle de Casos de Uso

Nombre:	Acceder al Sistema
Actores:	Coordinador de Microcrédito, Asesor de Microcrédito
Función:	Permitir acceso al sistema
Descripción:	Existen Usuarios registrados, solo estos tienen acceso al sistema. Se presenta pantalla de Inicio de sesión. Si el acceso es exitoso muestra ventana de selección de rol. Luego se carga la ventana de inicio y el árbol de procesos según el rol escogido.

Nombre:	Parametrizar Datos Generales
Actores:	Coordinador de Microcrédito
Función:	Parametrizar y dar mantenimiento a tablas de datos como tipos, numeraciones, etc. que son utilizadas por el sistema para cargar listas de valores.
Descripción:	Deben existir formularios que permitan dar mantenimiento a tablas comunes de la aplicación. Son formularios que permitan hacer tanto ingreso, modificación y permitan la opción de caducar registro, ya que la arquitectura del sistema no debe permitir eliminación de registros. Existen scripts SQL para parametrizar las tablas cuya información no cambia para evitar ingreso manual de los datos.

Nombre:	Crear Usuarios
Actores:	Coordinador de Microcrédito
Función:	Proceso de creación de Usuarios
Descripción:	El Administrador es el responsable de crear nuevos usuarios, generalmente de tipo Asesor. Debe existir un script SQL que parametrice usuarios iniciales.

Nombre:	Asignar Contraseñas
Actores:	Coordinador de Microcrédito
Función:	Creación de contraseñas para usuarios

Descripción:	El Coordinador tiene que asignar contraseñas a los usuarios para garantizar su acceso. También puede resetear las contraseñas de los usuarios en caso de ser necesario. Existe una contraseña por defecto almacenada en la BD la cual se toma como referencia.

Nombre:	Asignar Zonas Geográficas
Actores:	Coordinador de Microcrédito
Función:	Asignación de zonas geográficas a Asesores
Descripción:	Después de crear zonas geográficas el coordinador utiliza una transacción para asignar estas a los asesores de microcrédito

Nombre:	Crear Productos de Microcrédito
Actores:	Coordinador de Microcrédito
Función:	Ingreso de Productos a ser comercializados
Descripción:	Se definen los productos a ser comercializados, sus montos, periodos mínimos y máximos en que se realizan los préstamos y la tasa de interés definida según especifica El Banco Central del Ecuador. Una vez creados los productos se deben asignar a los asesores para que estos tengan acceso a su promoción en etapas posteriores.

Nombre:	Crear Zonas Geográficas
Actores:	Coordinador de Microcrédito
Función:	Creación y modificación de Zonas Geográficas
Descripción:	El coordinador puede ingresar zonas geográficas a través de coordenadas en un mapa. Las zonas pueden ser de tres tipos: puntos, rutas y polígonos. Es posible modificar las zonas en caso de que el coordinador requiera hacerlo.

Nombre:	Consultar Zonas Geográficas
Actores:	Asesor de Microcrédito
Función:	Consulta de zonas geográficas asignadas
Descripción:	El Asesor accede al sistema y puede ver un listado de todas las zonas asignadas a él.

	Una ventana muestra un mapa donde se visualizan puntos, rutas o polígonos asignados según fueron asignados por el coordinador de microcrédito.

Nombre:	Cambiar Contraseña
Actores:	Asesor de Microcrédito
Función:	Cambio de contraseña
Descripción:	El coordinador asigna a los usuarios una contraseña por defecto, cuando el usuario accede puede cambiar su contraseña por la que este desee.

Nombre:	Crear Socios Individuales
Actores:	Asesor de Microcrédito
Función:	Ingreso y modificación de socios individuales.
Descripción:	El Asesor puede crear socios o clientes individuales por medio de un formulario donde debe especificar: Datos de la persona, nombre del asesor responsable del socio, la frecuencia de reunión, el día de reunión y las actividades a las que se dedica el socio de manera descriptiva.

Nombre:	Crear Socios Grupales
Actores:	Asesor de Microcrédito
Función:	Ingreso y modificación de grupos y socios grupales.
Descripción:	El Asesor primero crea al grupo por medio de un formulario donde debe especificar: Nombre del asesor responsable del grupo, la frecuencia de reunión, el día de reunión, la fecha de creación, actividades a las que se dedica el grupo de manera descriptiva y una descripción del grupo para identificarlo. Junto con la información del grupo, se asignan socios al mismo. Entre 3 y 15 exactamente. Se asigna a uno de ellos como presidente, otro secretario y los demás únicamente como miembros.

Nombre:	Crear Solicitud
Actores:	Asesor de Microcrédito
Función:	Creación de Solicitud de microcrédito

Descripción:	El Asesor coloca la información recopilada en los módulos anteriores sobre datos del cliente, del asesor que ingresa la solicitud y datos principalmente de carácter financiero tales como montos de préstamo, plazos, frecuencia de pago y destino de fondos. La Aprobación es inmediata y es responsabilidad del Asesor, por lo tanto en este mismo proceso se coloca el estado de la solicitud.

Nombre:	Registrar Recomendación
Actores:	Asesor de Microcrédito
Función:	Ingreso de recomendación
Descripción:	El Asesor es responsable de ingresar datos que de alguna forma garanticen que un préstamo pueda ser recuperado, para ello registra información de carácter descriptivo de: Documentos, Unidad Económica, Unidad Familiar, Moral de Pago, Historial de crédito y finalmente la recomendación.

Nombre:	Consultar Solicitudes
Actores:	Asesor de Microcrédito
Función:	Consulta de Solicitudes
Descripción:	El Asesor lista todas las solicitudes por estado e información financiera.

Nombre:	Instrumentar
Actores:	Asesor de Microcrédito
Función:	Crear cuentas, aprobar y generar tablas de cuotas a las solicitudes en lote.
Descripción:	Este proceso se realiza automáticamente por medio del emulador de Core bancario, el Asesor lista las solicitudes ingresadas y tiene la opción de realizar la instrumentación de los prestamos.

Nombre:	Consultar Cuotas
Actores:	Asesor de Microcrédito
Función:	Consulta de Cuotas
Descripción:	El Asesor lista la tabla de cuotas generada en la instrumentación y sirve para llevar un reporte de estado de fechas de cobro y de visita a los socios.

Nombre:	Recaudar Cuotas
Actores:	Asesor de Microcrédito
Función:	Cobrar cuotas de un préstamo
Descripción:	Tal como el listado de cuotas, en este proceso se listan las cuotas según numero de solicitud con una opción para pagarla. Los pagos se realizan en la fecha máxima de pago y solo puede hacerse el pago de una cuota a la vez. Si el socio desea cancelar el monto del préstamo completo o restante tiene que hacerlo por medio de la entidad Bancaria directamente.

Nombre:	Verificar destino de Fondos
Actores:	Asesor de Microcrédito
Función:	Ingreso de datos del destino de Fondos de cada solicitud
Descripción:	El Asesor visita periódicamente a los socios para verificar que el préstamo haya sido utilizado para el fin que fue solicitado. Ingresa información descriptiva que corrobore esto y le permita llevar un control permanente.

Nombre:	Consultar Movimientos del día
Actores:	Asesor de Microcrédito
Función:	Consulta de movimientos del día
Descripción:	El Asesor lista las cuotas recaudadas en el día dado que comisiona por cada una que es recaudada, por lo tanto ese listado le permite llevar control de sus ganancias.

ANEXO 2. Procesos de la Aplicación.

Sub.	Mod.	Pro.	Título	Descripción
GENERALES				
G	00	-	Menú	
G	00	1	Menú principal	Proceso de Generación del Menú principal
G	01	-	Parámetros	
G	01	1	Parámetros generales	Definición de Parámetros utilizados por el sistema
G	01	2	Consulta combo	Proceso que ejecuta la consulta en una Lista de Valores
G	02	-	Listas de valores (Combos)	
G	02	1	Listas de valores para los combos	Proceso que permite mostrar las Listas de Valores
G	03	-	Localización	
G	03	1	Países	Proceso para mantenimiento de Países
G	03	2	Provincias	Proceso para mantenimiento de Provincias
G	03	3	Cantones	Proceso para mantenimiento de Cantones
G	03	4	Parroquias	Proceso para mantenimiento de Parroquias
SEGURIDADES				
A	00	-	Autenticación	
A	00	1	Login	Proceso que muestra el Login al sistema
A	00	2	Logout	Proceso que ejecuta el Logout del sistema
A	01	-	Datos generales	
A	01	1	Estatus De Usuarios	Manejo de estatus de usuario
A	01	2	Tipos De Usuario	Manejo de tipos de usuario
A	01	3	Subsistemas	Manejo de Subsistemas
A	01	4	Módulos	Manejo de Módulos
A	01	5	Procesos	Manejo de Procesos
A	01	6	Componentes	Manejo de Componentes
A	01	7	Componentes por Proceso	Asignación de Componentes a Procesos
A	02	-	Usuarios y Roles	
A	02	1	Roles	Manejo de Roles
A	02	2	Procesos por Rol	Asignación de Procesos a Roles
A	02	3	Usuarios	Creación de Usuarios
A	02	4	Roles por Usuario	Asignación de Roles a Usuarios
A	02	5	Cambio Contraseña (Usuarios)	Cambio de contraseña
A	02	6	Reseteo Contraseña (Admin)	Reseteo Contraseña
A	02	7	Terminales	Manejo de Terminales

PERSONAS				
B	00	-	Parametrizacion	
B	00	1	Tipos de persona	Manejo de tipos de personas
B	00	2	Tipos de identificación	Manejo de tipos de identificaciones
B	00	3	Géneros	Manejo de Géneros
B	00	4	Estados civiles	Manejo de estados civiles
B	00	5	Profesiones	Manejo de profesiones
B	00	6	Tipos de dirección	Manejo de tipos de direcciones
B	00	7	Tipos de teléfono	Manejo de tipos de teléfonos
B	01	-	Personas naturales	
B	01	1	Personas naturales	Ingreso de Personas Naturales
B	01	2	Direcciones	Ingreso de Direcciones para una persona
B	01	3	Teléfonos	Ingreso de Teléfonos para una persona
MICROCRÉDITOS				
C	00	-	Parametrización	
C	00	1	Monedas	Manejo de monedas
C	00	2	Estatus de solicitud	Manejo de estatus de solicitud
C	00	3	Tipos de Cuota	Manejo de tipos de cuota
C	00	4	Frecuencias	Manejo de frecuencias (para visitas y pagos)
C	00	5	Destinos de fondos	Manejo de destinos de fondos
C	01	-	Planificación	
C	01	1	Asesor de Microcrédito	Listado de Asesores registrados en el sistema
C	01	2	Zonas Geográficas	Mantenimiento de zonas con manejo de mapas
C	01	3	Zonas por Asesor	Asignación de zonas geográficas a cada asesor
C	01	4	Definición de productos de microcrédito	Creación de productos de microcrédito
C	01	6	Zonas Asignadas	Accesible solo por los Asesores para visualizar las zonas que le fueron asignadas
C	02	-	Comercialización	
C	02	1	Clientes individuales	Creación de clientes Individuales
C	02	2	Clientes grupales	Creación de grupos y asignación de miembros a estos.
C	03	-	Solicitud	
C	03	1	Solicitud de microcrédito / Représtamo.	Ingreso de Solicitudes. Aprobación/Denegación inmediata
C	03	2	Recomendación	Proceso de registro de Recomendación dada por el asesor
C	04	-	Instrumentación Core	
C	04	1	Consulta de solicitudes	Mostrar un listado de las solicitudes y sus estados. Útil previo a la instrumentación.
C	04	2	Instrumentación en lote	Instrumentación empleando el Simulador de Core Bancario.
C	05	-	Seguimiento y recuperación	

C	05	1	Verificación de destinos de fondos	Proceso para verificación de destino de fondos
C	05	2	Consulta de cuotas	Listado de cuotas según número de solicitud
C	05	3	Pago de cuotas	Listado de cuotas con opción de pago

ANEXO 3. Diagramas de componentes

Diagrama de componentes a nivel de módulos de la aplicación

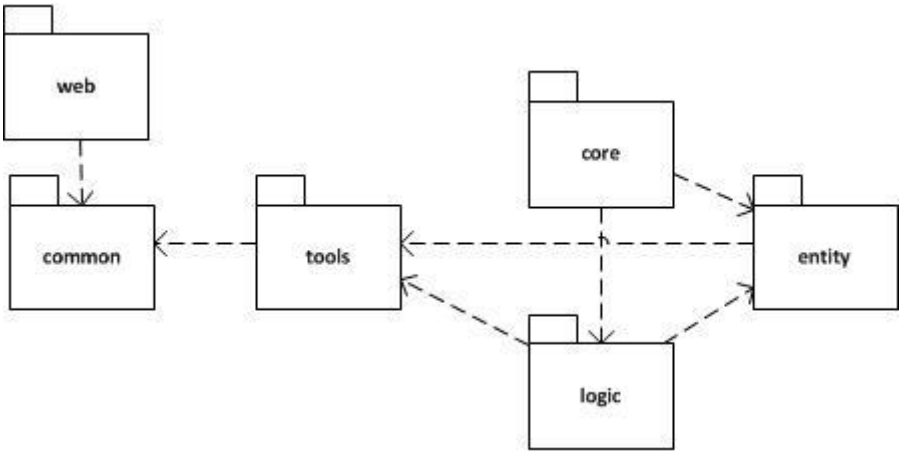
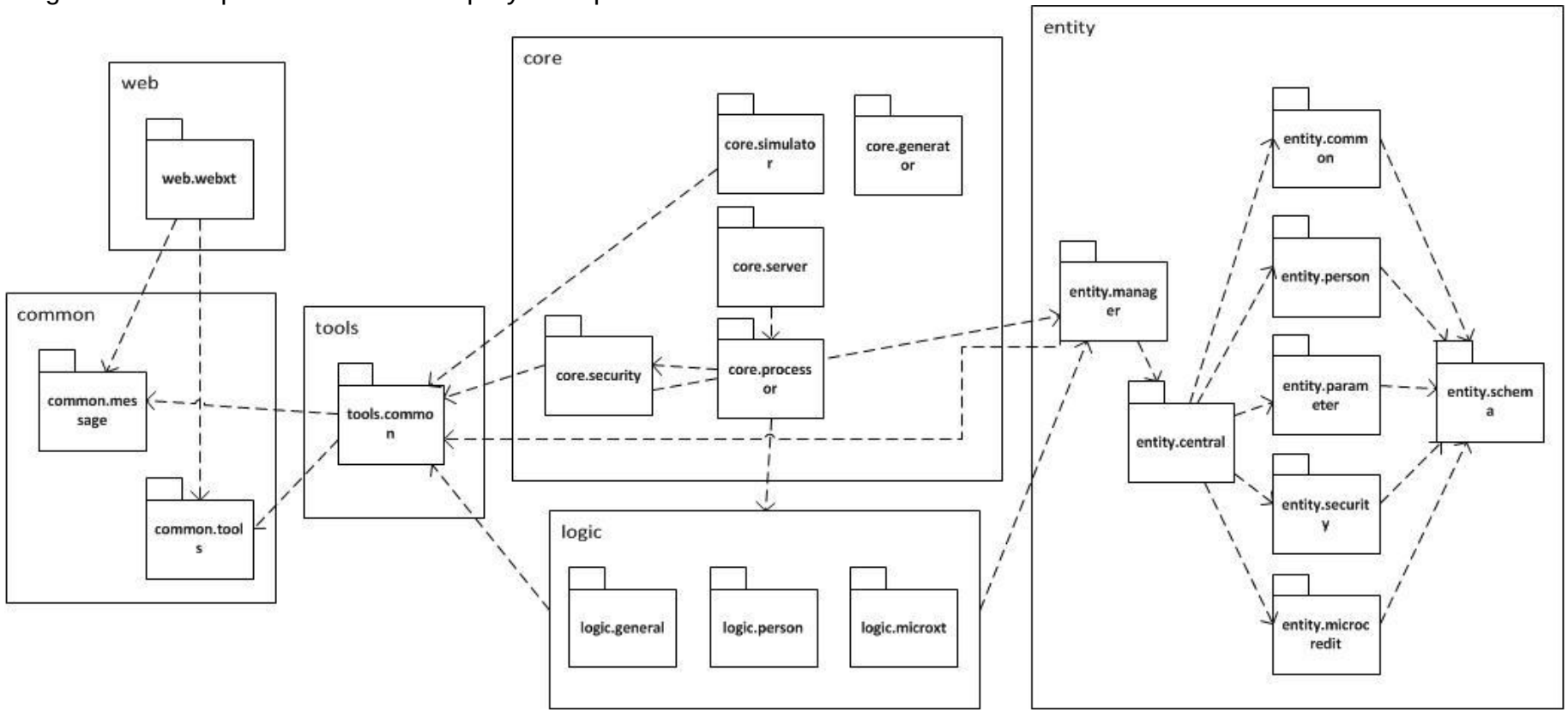
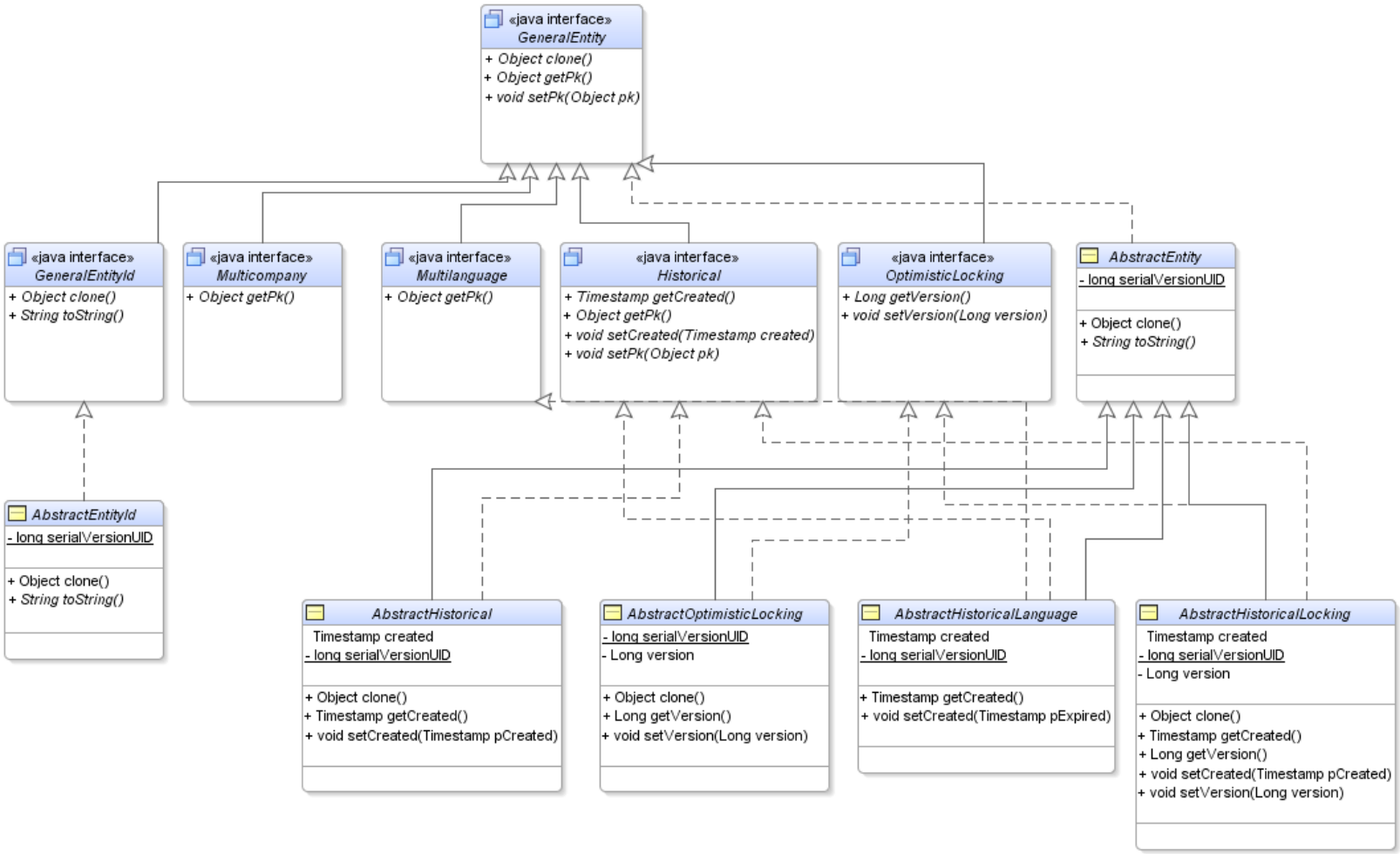


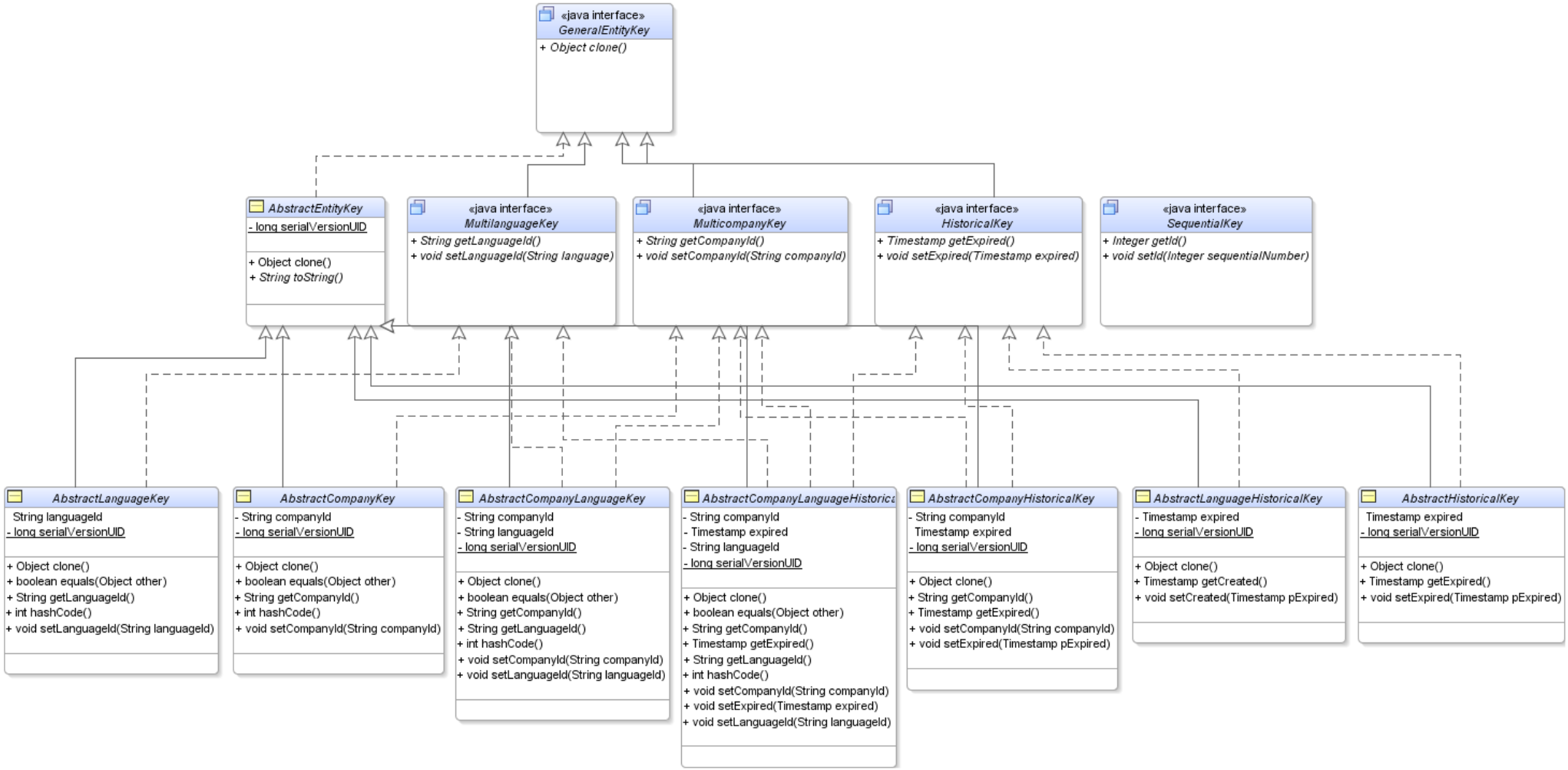
Diagrama de componentes a nivel de proyectos por módulo



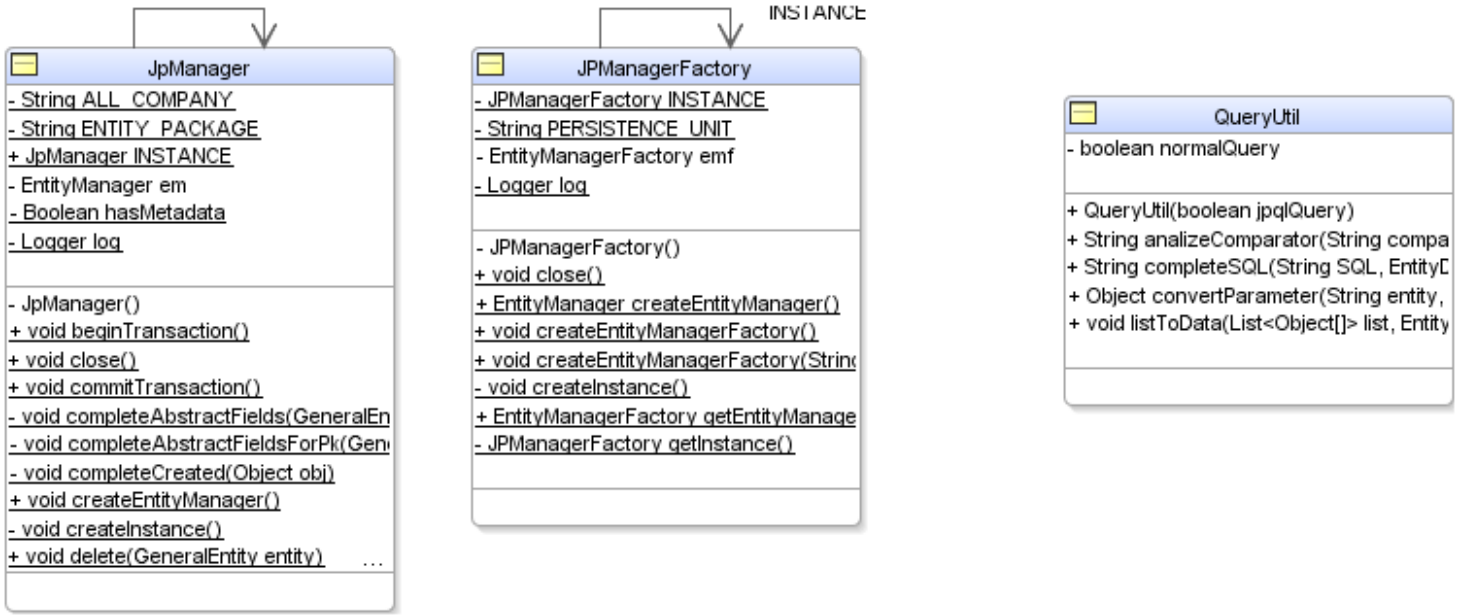
ANEXO 4. Diagramas de clase

Proyecto: entity.schema

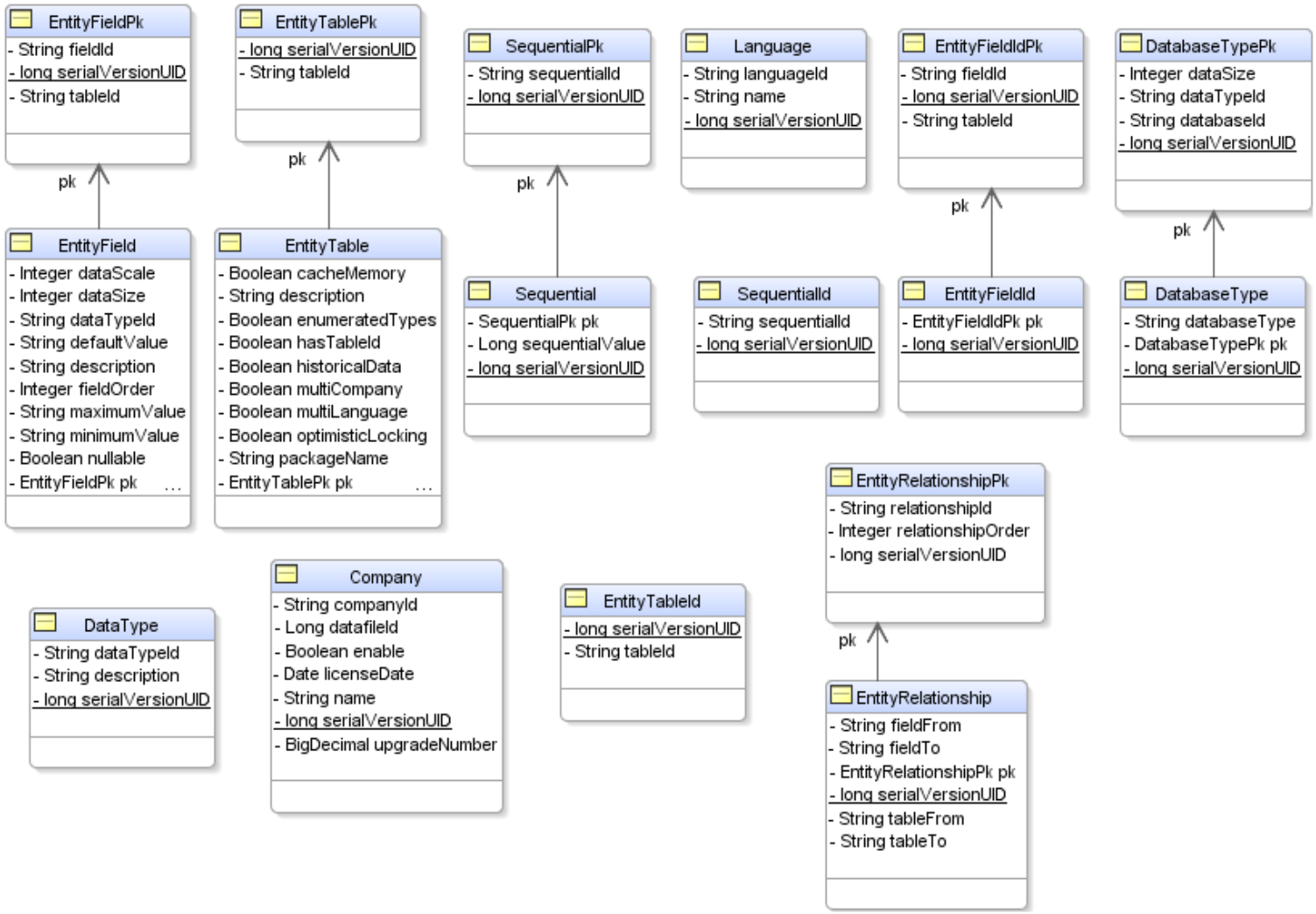



«java interface»
GeneralEntityKey
+ Object clone()
AbstractEntityKey
- long serialVersionUID
+ Object clone()
+ String toString()
«java interface»
MultilanguageKey
+ String getLanguageId()
+ void setLanguageId(String language)
«java interface»
MulticompanyKey
+ String getCompanyId()
+ void setCompanyId(String companyId)
«java interface»
HistoricalKey
+ Timestamp getExpired()
+ void setExpired(Timestamp expired)
«java interface»
SequentialKey
+ Integer getId()
+ void setId(Integer sequentialNumber)
AbstractLanguageKey
String languageId
- long serialVersionUID
+ Object clone()
+ boolean equals(Object other)
+ String getLanguageId()
+ int hashCode()
+ void setLanguageId(String languageId)
AbstractCompanyKey
- String companyId
- long serialVersionUID
+ Object clone()
+ boolean equals(Object other)
+ String getCompanyId()
+ int hashCode()
+ void setCompanyId(String companyId)
AbstractCompanyLanguageKey
- String companyId
- String languageId
- long serialVersionUID
+ Object clone()
+ boolean equals(Object other)
+ String getCompanyId()
+ String getLanguageId()
+ int hashCode()
+ void setCompanyId(String companyId)
+ void setLanguageId(String languageId)
AbstractCompanyLanguageHistoric
- String companyId
- Timestamp expired
- String languageId
- long serialVersionUID
+ Object clone()
+ boolean equals(Object other)
+ String getCompanyId()
+ Timestamp getExpired()
+ String getLanguageId()
+ int hashCode()
+ void setCompanyId(String companyId)
+ void setExpired(Timestamp expired)
+ void setLanguageId(String languageId)
AbstractCompanyHistoricalKey
- String companyId
Timestamp expired
- long serialVersionUID
+ Object clone()
+ String getCompanyId()
+ Timestamp getExpired()
+ void setCompanyId(String companyId)
+ void setExpired(Timestamp pExpired)
AbstractLanguageHistoricalKey
- Timestamp expired
- long serialVersionUID
+ Object clone()
+ Timestamp getCreated()
+ void setCreated(Timestamp pExpired)
AbstractHistoricalKey
Timestamp expired
- long serialVersionUID
+ Object clone()
+ Timestamp getExpired()
+ void setExpired(Timestamp pExpired)

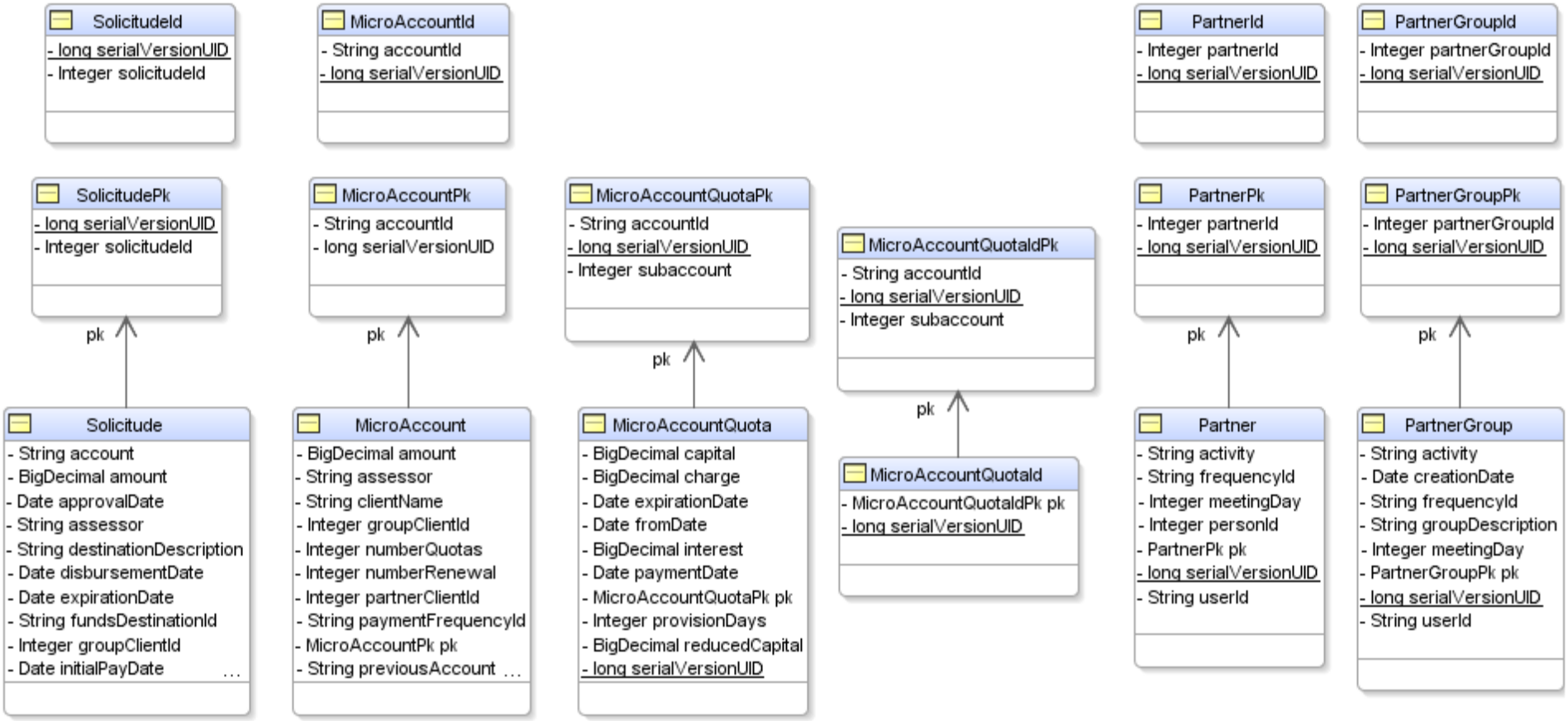
Proyecto: entity.manager

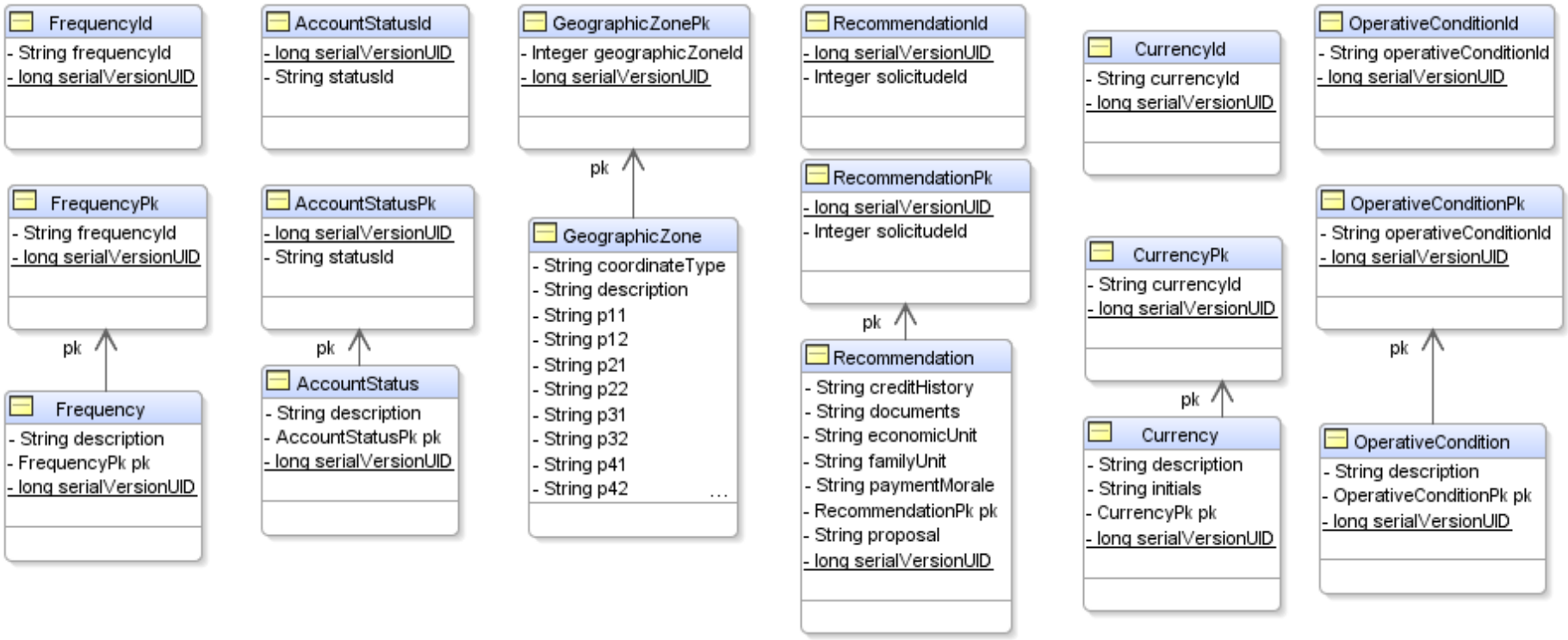


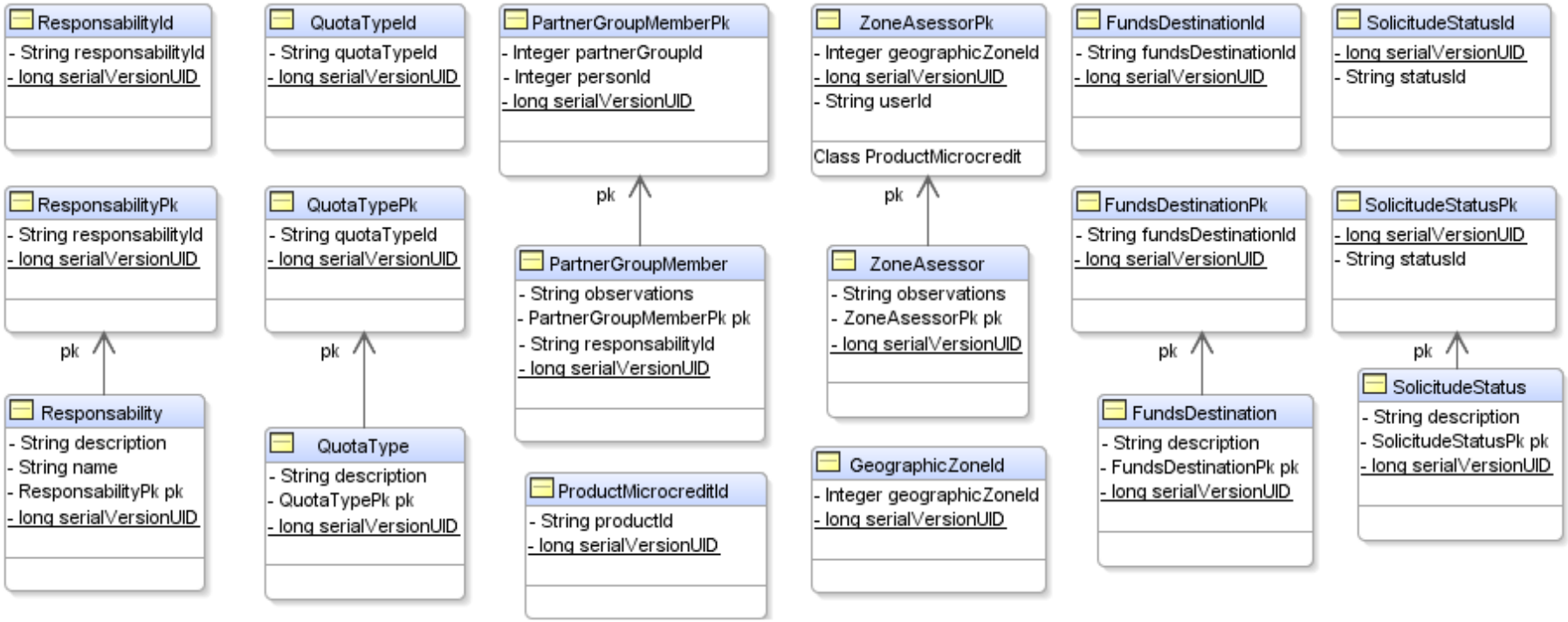
Proyecto: entity.common



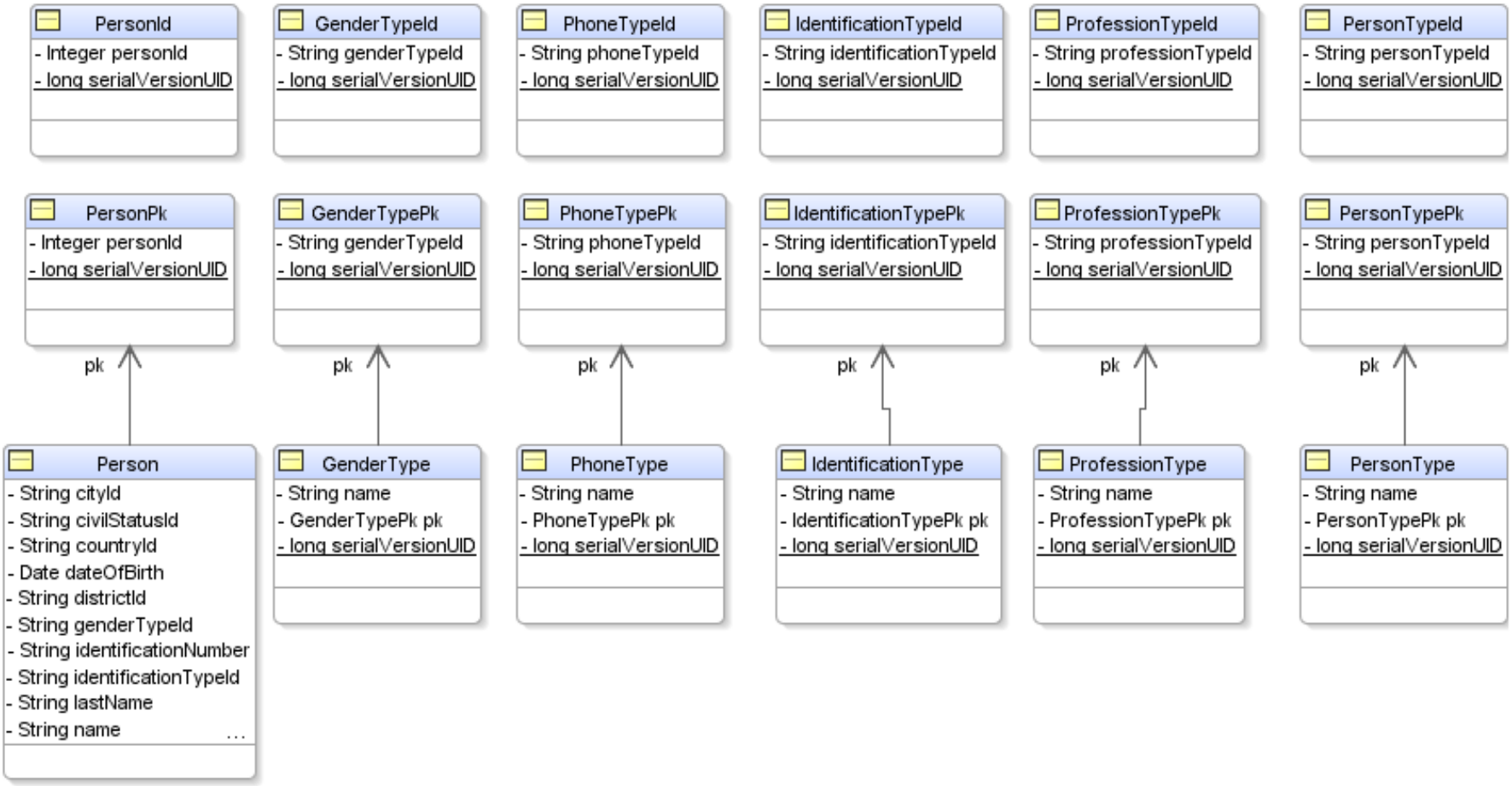
Proyecto: entity.microcredit

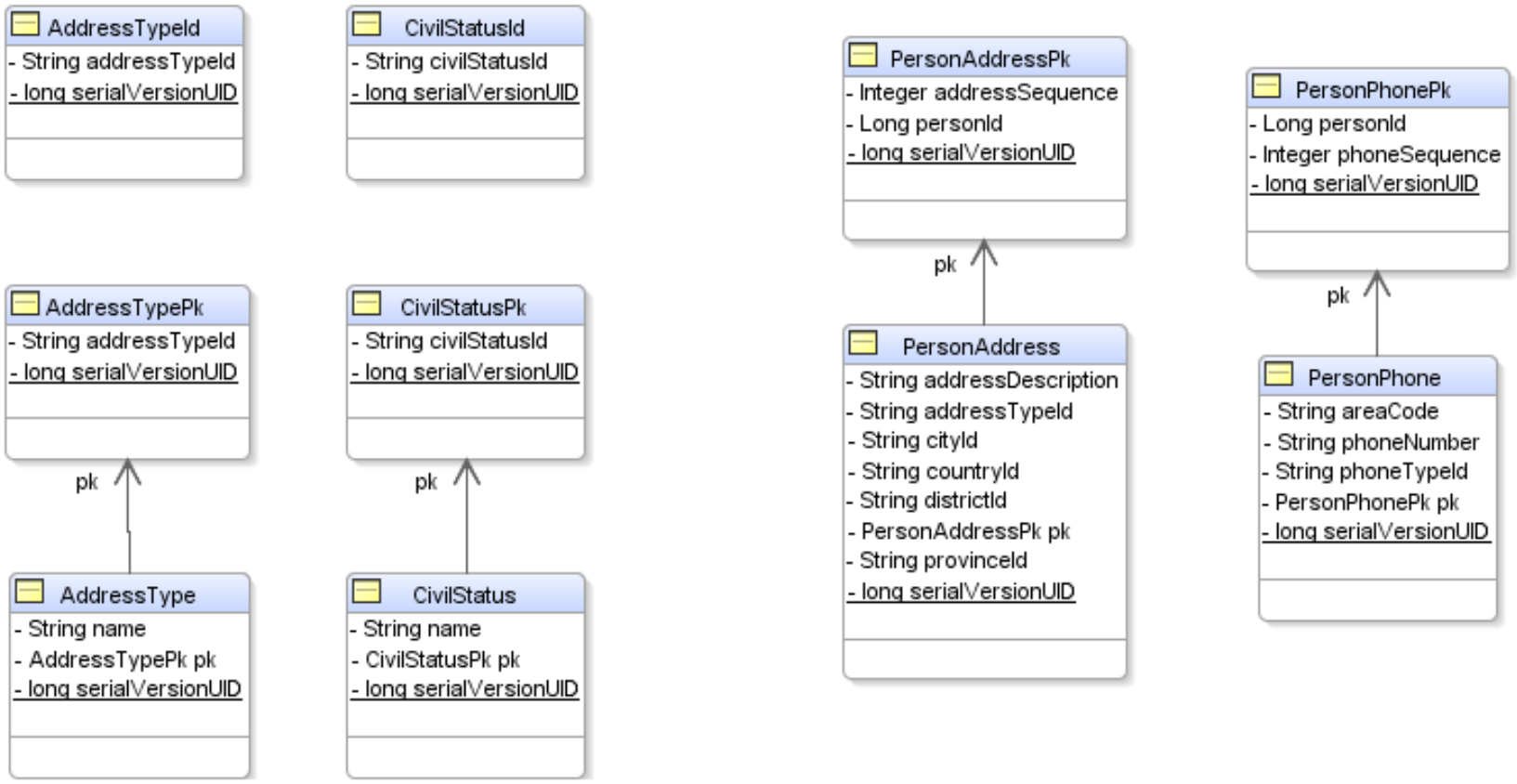



FrequencyId
- String frequencyId
- long serialVersionUID
FrequencyPk
- String frequencyId
- long serialVersionUID
pk
Frequency
- String description
- FrequencyPk pk
- long serialVersionUID
AccountStatusId
- long serialVersionUID
- String statusId
AccountStatusPk
- long serialVersionUID
- String statusId
pk
AccountStatus
- String description
- AccountStatusPk pk
- long serialVersionUID
GeographicZonePk
- Integer geographicZoneId
- long serialVersionUID
pk
GeographicZone
- String coordinateType
- String description
- String p11
- String p12
- String p21
- String p22
- String p31
- String p32
- String p41
- String p42
...
RecommendationId
- long serialVersionUID
- Integer solicitudeId
RecommendationPk
- long serialVersionUID
- Integer solicitudeId
pk
Recommendation
- String creditHistory
- String documents
- String economicUnit
- String familyUnit
- String paymentMorale
- RecommendationPk pk
- String proposal
- long serialVersionUID
CurrencyId
- String currencyId
- long serialVersionUID
CurrencyPk
- String currencyId
- long serialVersionUID
pk
Currency
- String description
- String initials
- CurrencyPk pk
- long serialVersionUID
OperativeConditionId
- String operativeConditionId
- long serialVersionUID
OperativeConditionPk
- String operativeConditionId
- long serialVersionUID
pk
OperativeCondition
- String description
- OperativeConditionPk pk
- long serialVersionUID

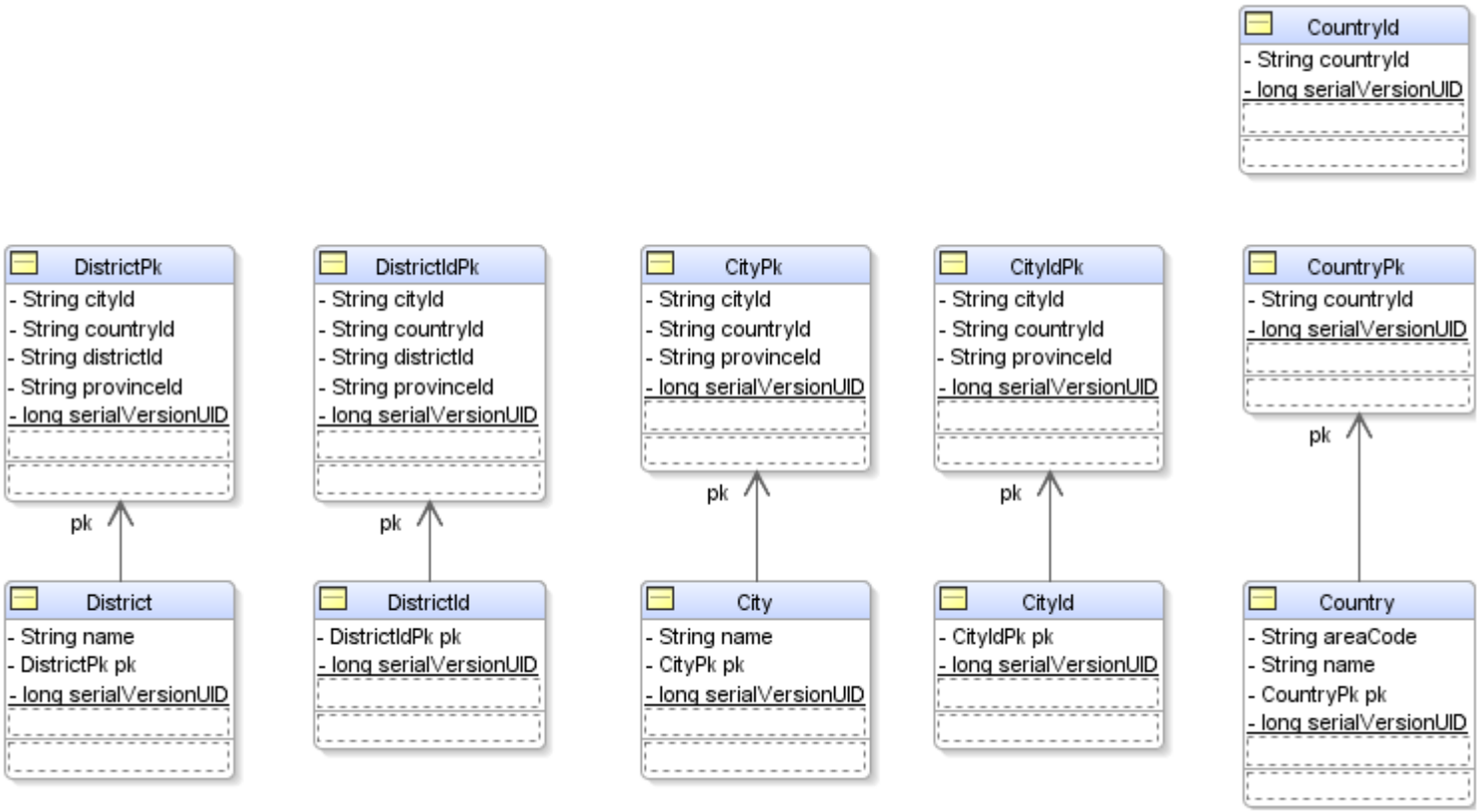

ResponsabilityId
- String responsabilityId
- long serialVersionUID
QuotaTypeId
- String quotaTypeId
- long serialVersionUID
PartnerGroupMemberPk
- Integer partnerGroupId
- Integer personId
- long serialVersionUID
ZoneAsessorPk
- Integer geographicZoneId
- long serialVersionUID
- String userId
Class ProductMicrocredit
FundsDestinationId
- String fundsDestinationId
- long serialVersionUID
SolicitudeStatusId
- long serialVersionUID
- String statusId
ResponsabilityPk
- String responsabilityId
- long serialVersionUID
QuotaTypePk
- String quotaTypeId
- long serialVersionUID
pk
PartnerGroupMember
- String observations
- PartnerGroupMemberPk pk
- String responsabilityId
- long serialVersionUID
pk
ZoneAsessor
- String observations
- ZoneAsessorPk pk
- long serialVersionUID
FundsDestinationPk
- String fundsDestinationId
- long serialVersionUID
SolicitudeStatusPk
- long serialVersionUID
- String statusId
pk
Responsability
- String description
- String name
- ResponsabilityPk pk
- long serialVersionUID
pk
QuotaType
- String description
- QuotaTypePk pk
- long serialVersionUID
ProductMicrocreditId
- String productId
- long serialVersionUID
GeographicZoneId
- Integer geographicZoneId
- long serialVersionUID
pk
FundsDestination
- String description
- FundsDestinationPk pk
- long serialVersionUID
pk
SolicitudeStatus
- String description
- SolicitudeStatusPk pk
- long serialVersionUID

Proyecto: entity.person

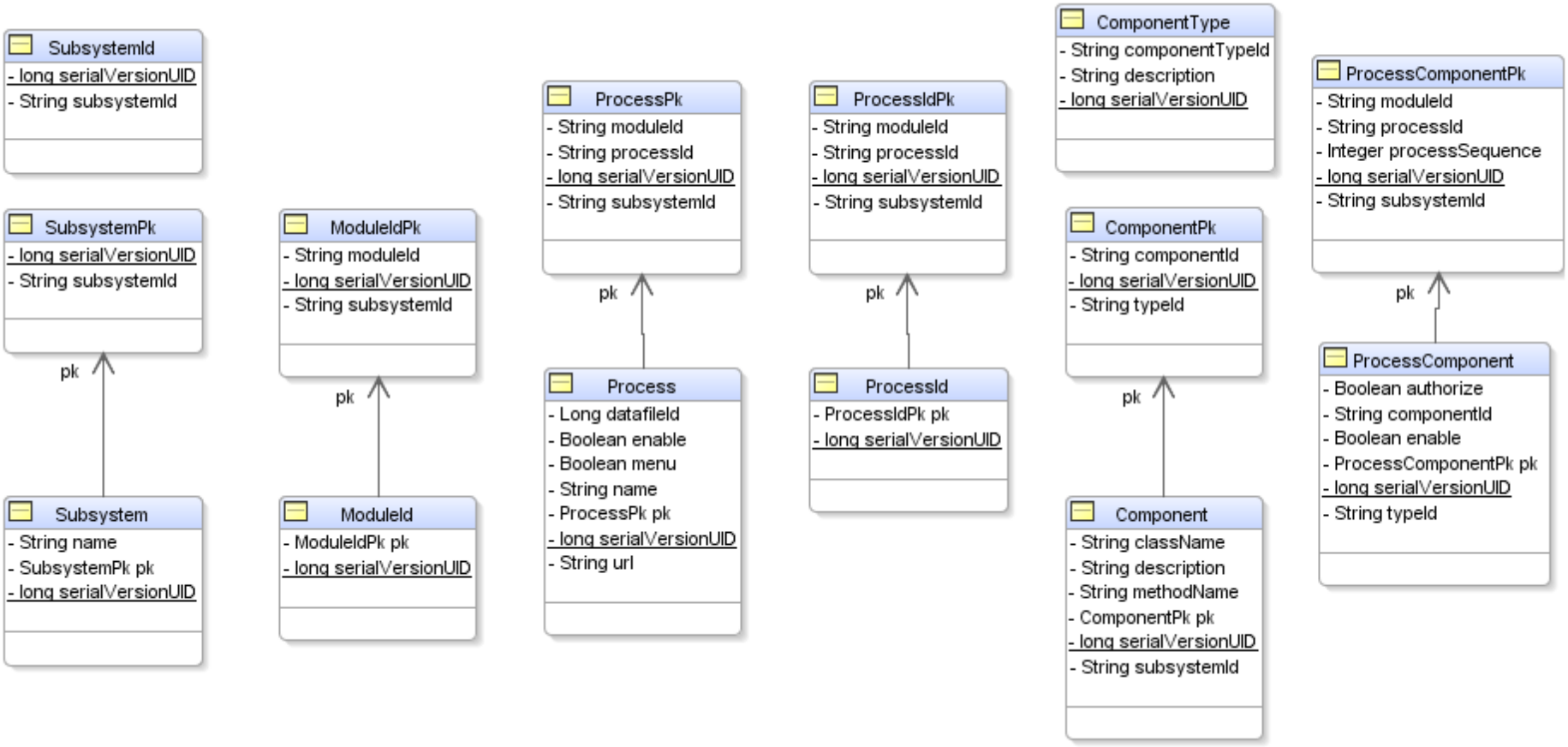


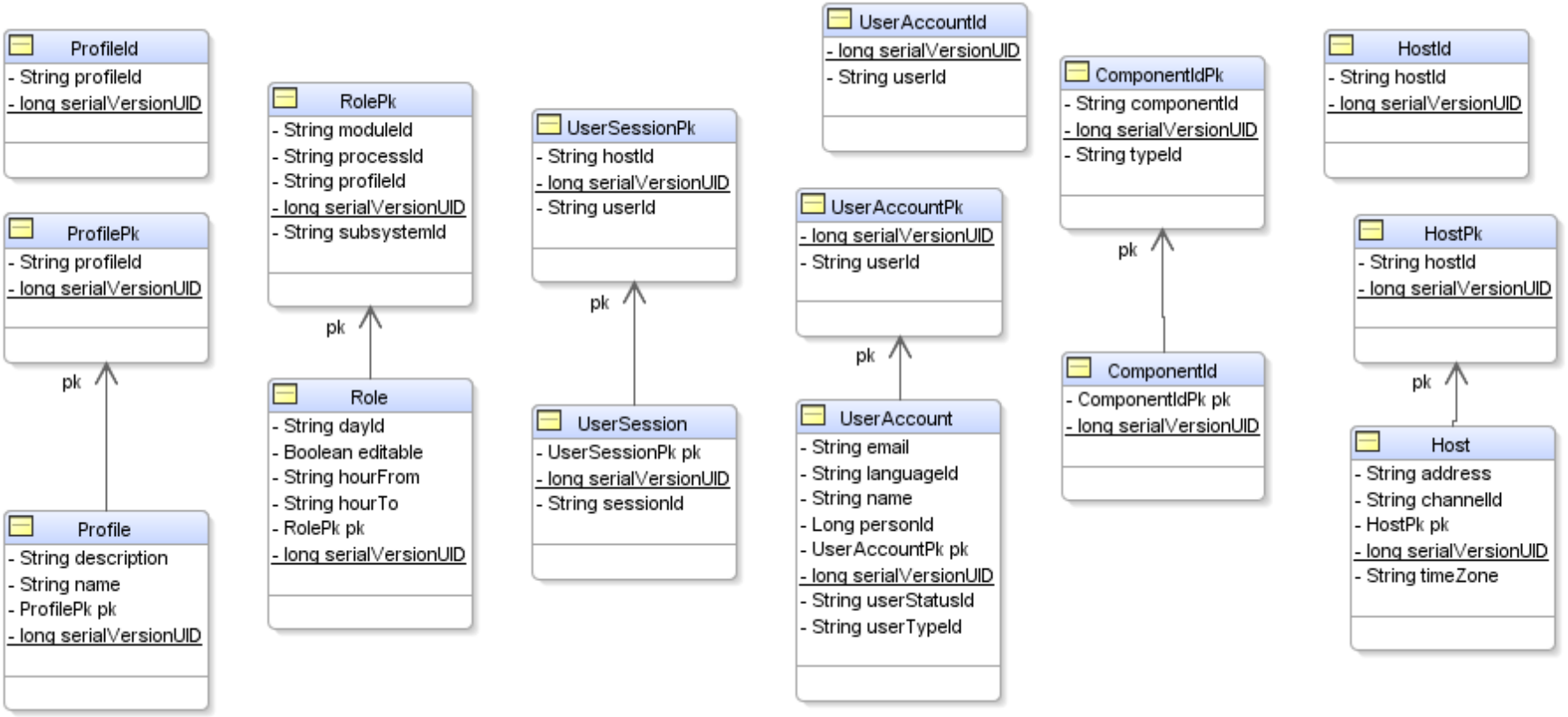

AddressTypeId
- String addressTypeId
- long serialVersionUID
CivilStatusId
- String civilStatusId
- long serialVersionUID
PersonAddressPk
- Integer addressSequence
- Long personId
- long serialVersionUID
PersonPhonePk
- Long personId
- Integer phoneSequence
- long serialVersionUID
AddressTypePk
- String addressTypeId
- long serialVersionUID
CivilStatusPk
- String civilStatusId
- long serialVersionUID
pk
pk
pk
pk
PersonAddress
- String addressDescription
- String addressTypeId
- String cityId
- String countryId
- String districtId
- PersonAddressPk pk
- String provinceId
- long serialVersionUID
PersonPhone
- String areaCode
- String phoneNumber
- String phoneTypeId
- PersonPhonePk pk
- long serialVersionUID
AddressType
- String name
- AddressTypePk pk
- long serialVersionUID
CivilStatus
- String name
- CivilStatusPk pk
- long serialVersionUID

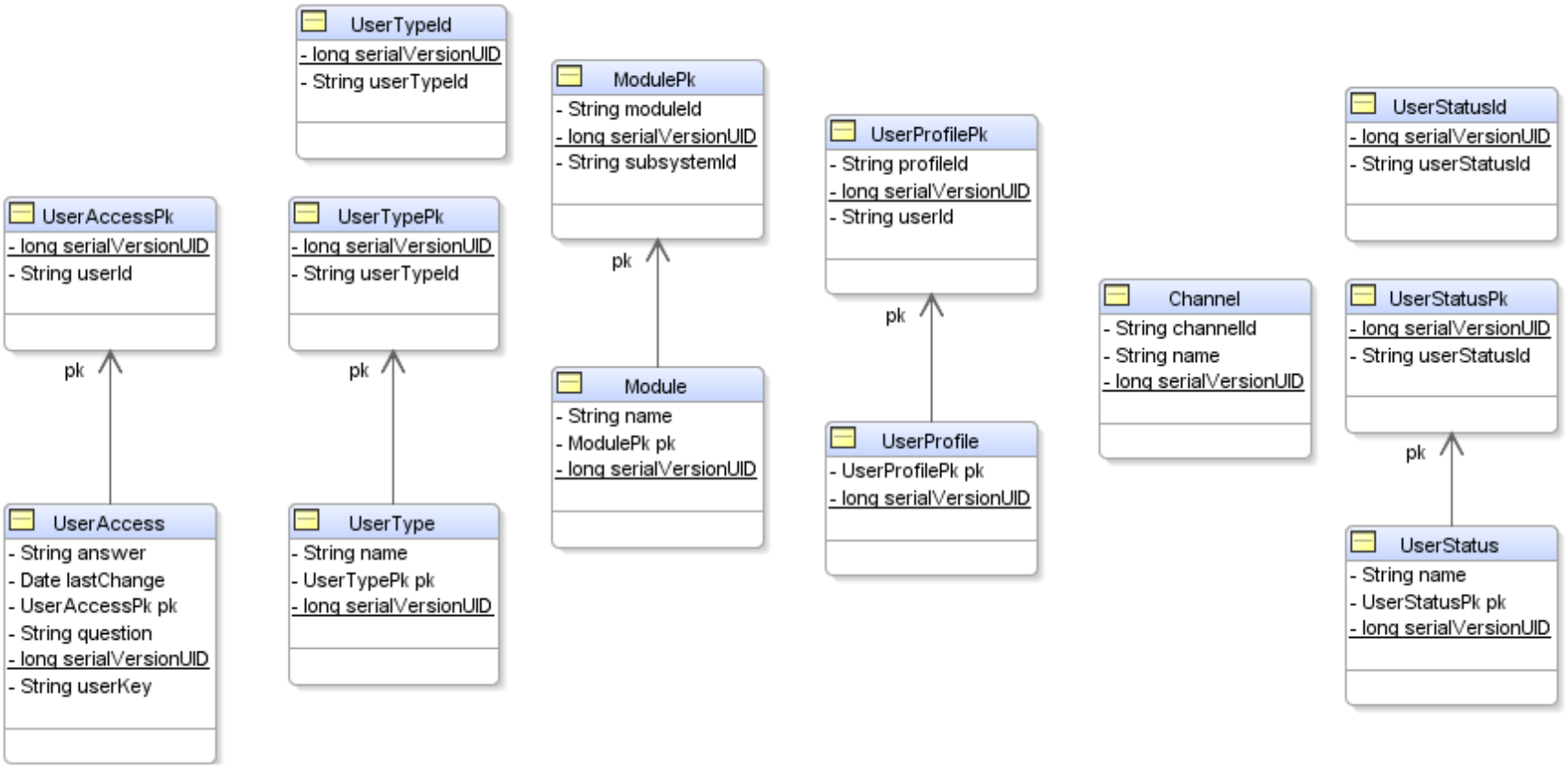
Proyecto: entity.parameter



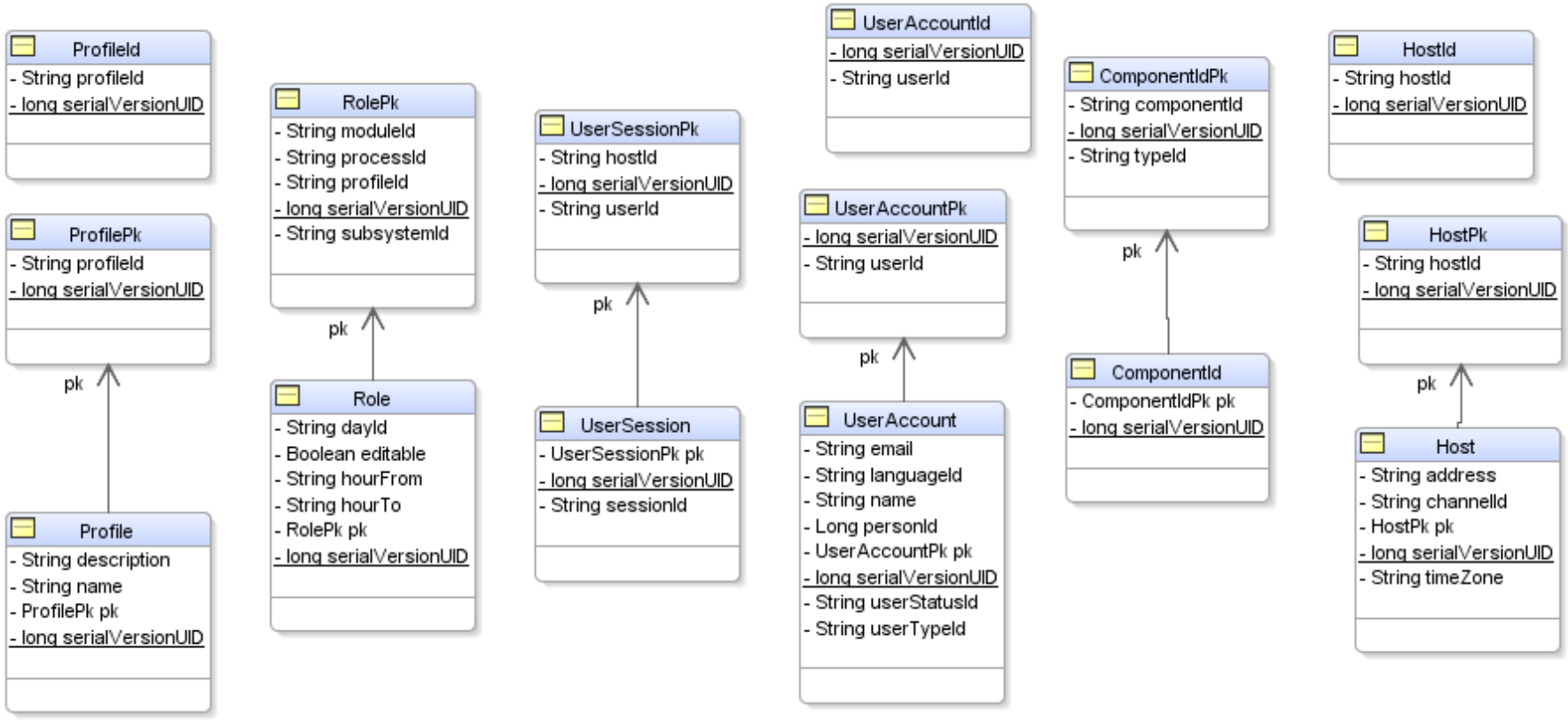
Proyecto: entity.security




ProfileId
- String profileId
- long serialVersionUID
ProfilePk
- String profileId
- long serialVersionUID
pk
Profile
- String description
- String name
- ProfilePk pk
- long serialVersionUID
RolePk
- String moduleId
- String processId
- String profileId
- long serialVersionUID
- String subsystemId
pk
Role
- String dayId
- Boolean editable
- String hourFrom
- String hourTo
- RolePk pk
- long serialVersionUID
UserSessionPk
- String hostId
- long serialVersionUID
- String userId
pk
UserSession
- UserSessionPk pk
- long serialVersionUID
- String sessionId
UserAccountId
- long serialVersionUID
- String userId
UserAccountPk
- long serialVersionUID
- String userId
pk
UserAccount
- String email
- String languageId
- String name
- Long personId
- UserAccountPk pk
- long serialVersionUID
- String userStatusId
- String userTypeId
ComponentIdPk
- String componentId
- long serialVersionUID
- String typeId
pk
ComponentId
- ComponentIdPk pk
- long serialVersionUID
HostId
- String hostId
- long serialVersionUID
HostPk
- String hostId
- long serialVersionUID
pk
Host
- String address
- String channelId
- HostPk pk
- long serialVersionUID
- String timeZone

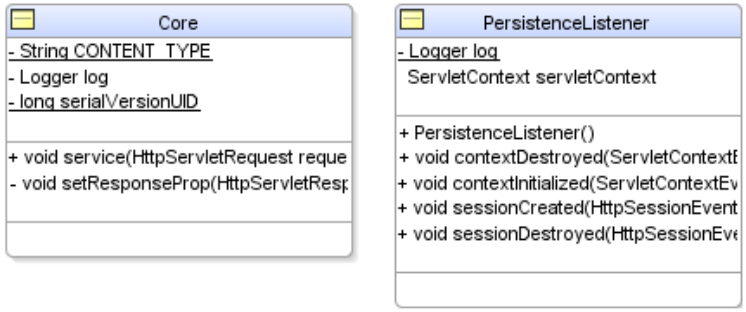

UserTypeId
- long serialVersionUID
- String userTypeId
ModulePk
- String moduleId
- long serialVersionUID
- String subsystemId
UserProfilePk
- String profileId
- long serialVersionUID
- String userId
UserStatusId
- long serialVersionUID
- String userStatusId
UserAccessPk
- long serialVersionUID
- String userId
UserTypePk
- long serialVersionUID
- String userTypeId
pk
pk
pk
pk
Channel
- String channelId
- String name
- long serialVersionUID
UserStatusPk
- long serialVersionUID
- String userStatusId
pk
Module
- String name
- ModulePk pk
- long serialVersionUID
UserProfile
- UserProfilePk pk
- long serialVersionUID
UserAccess
- String answer
- Date lastChange
- UserAccessPk pk
- String question
- long serialVersionUID
- String userKey
UserType
- String name
- UserTypePk pk
- long serialVersionUID
UserStatus
- String name
- UserStatusPk pk
- long serialVersionUID

Proyecto: core.generator

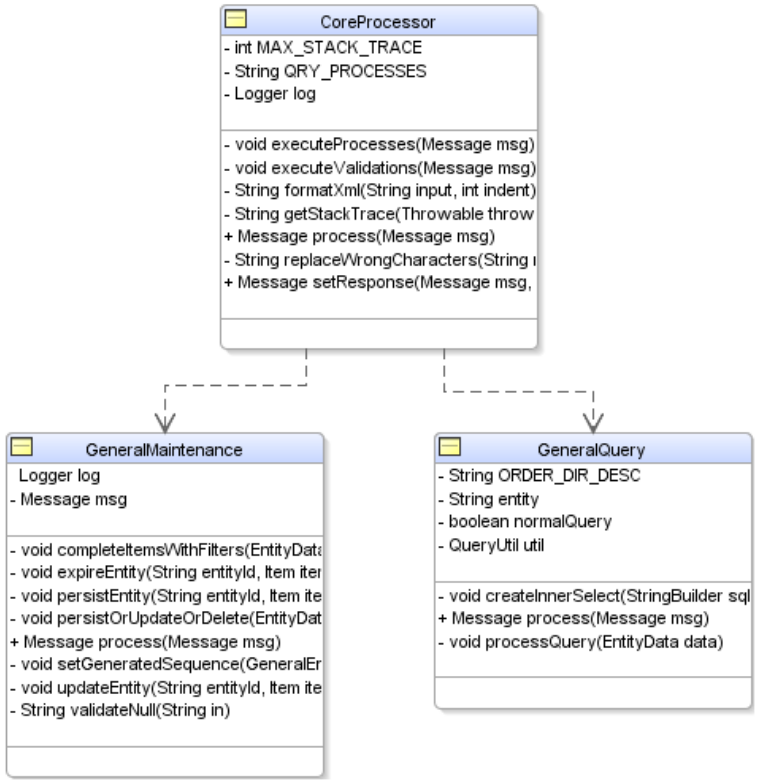


Proyectos: core.server / core.processor

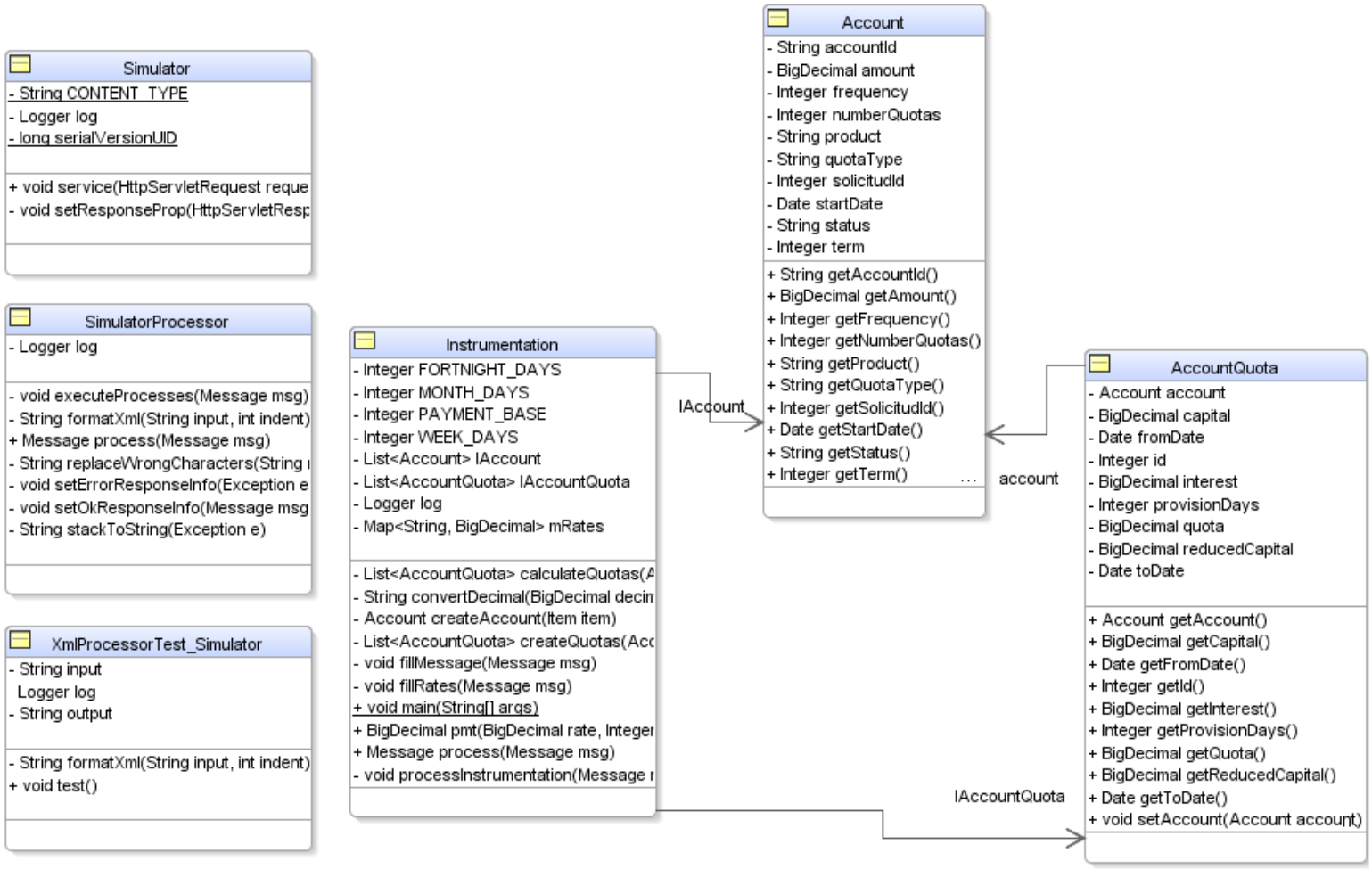
core.server



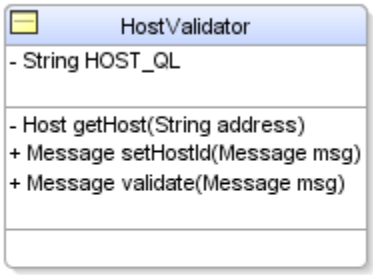
core.processor

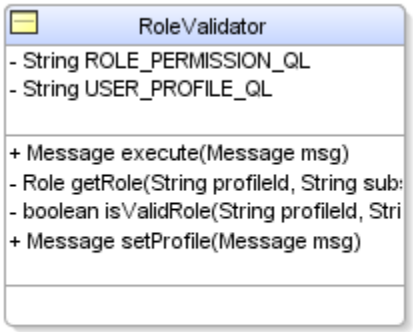


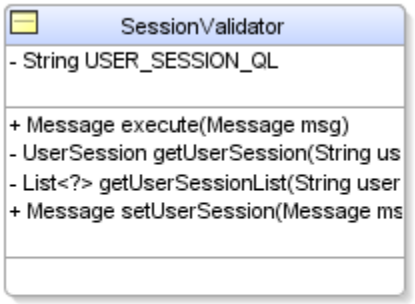
Proyecto: core.simulator

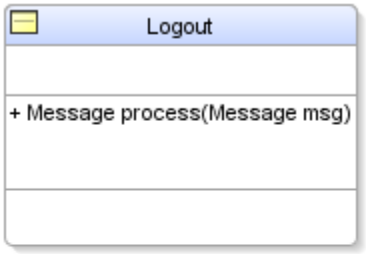


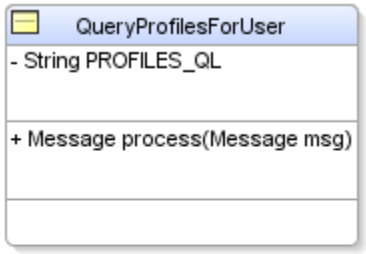
Proyecto: core.security

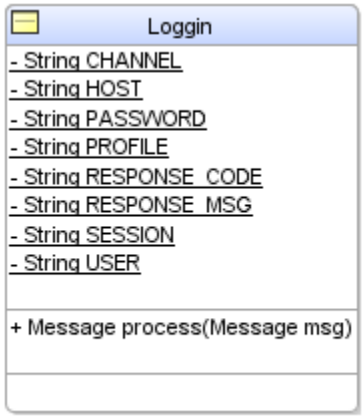






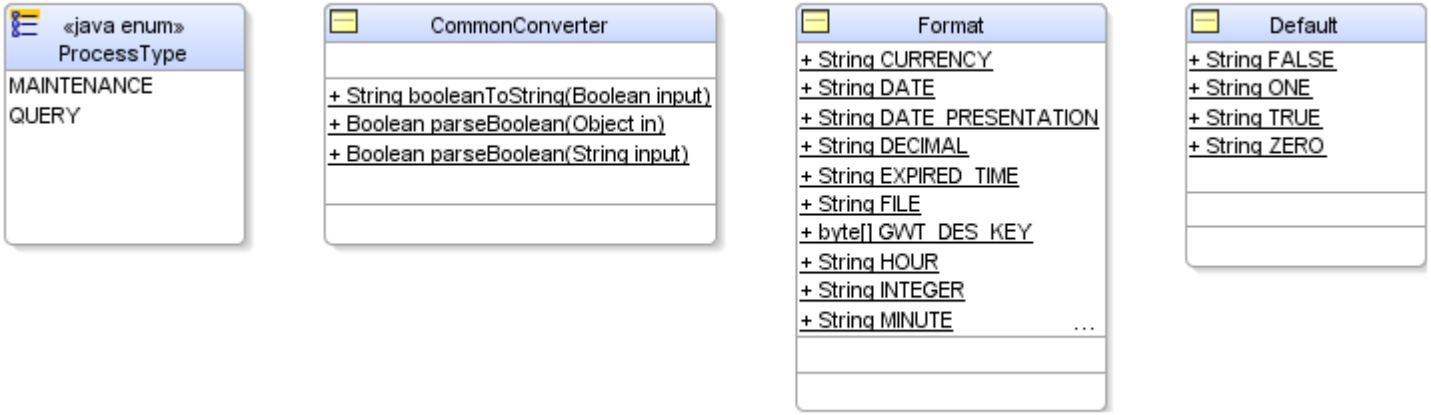




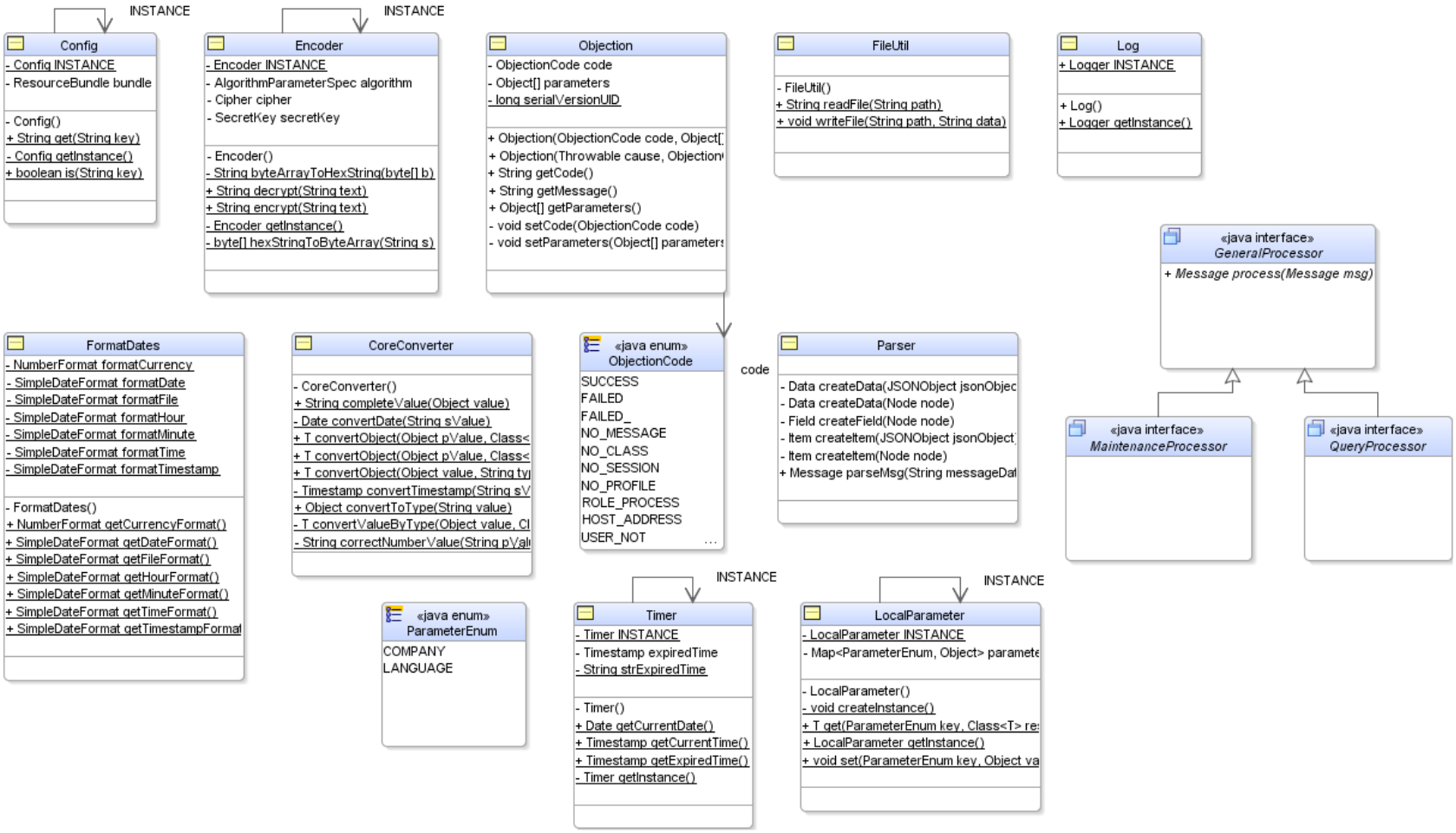


Proyecto: common.message

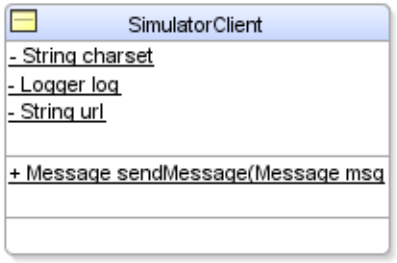
Proyecto: common.tools

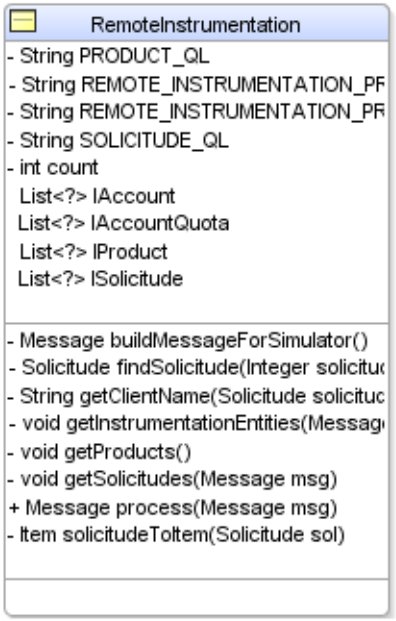

«java enum»
ProcessType
MAINTENANCE
QUERY
CommonConverter
+ String booleanToString(Boolean input)
+ Boolean parseBoolean(Object in)
+ Boolean parseBoolean(String input)
Format
+ String CURRENCY
+ String DATE
+ String DATE_PRESENTATION
+ String DECIMAL
+ String EXPIRED_TIME
+ String FILE
+ byte[] GWT_DES_KEY
+ String HOUR
+ String INTEGER
+ String MINUTE
...
Default
+ String FALSE
+ String ONE
+ String TRUE
+ String ZERO

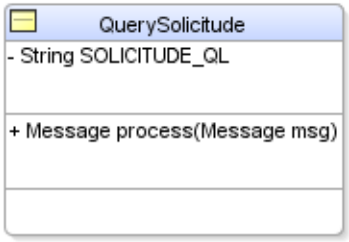
Proyecto: tolos.common



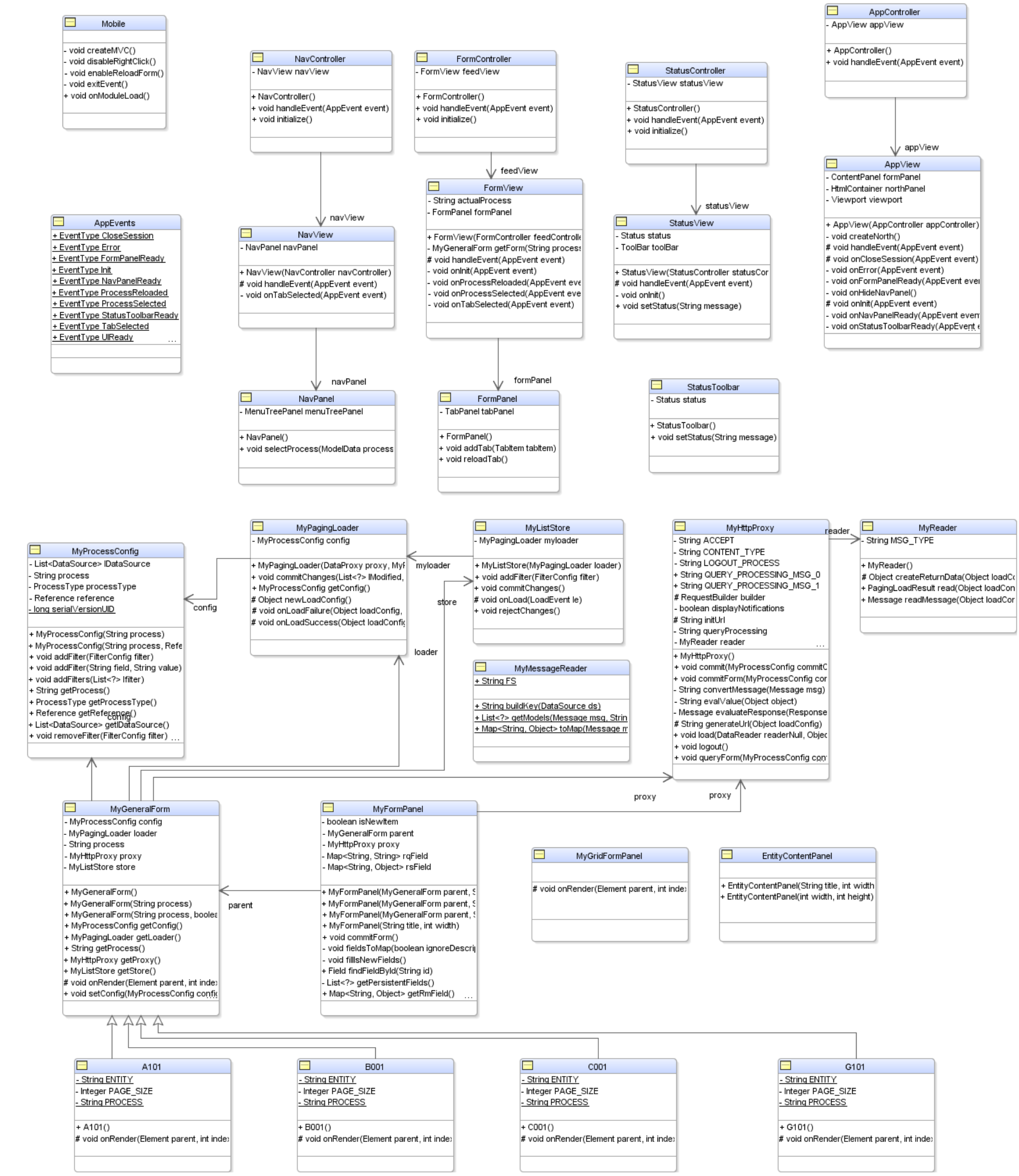
Proyecto: logic.microxt





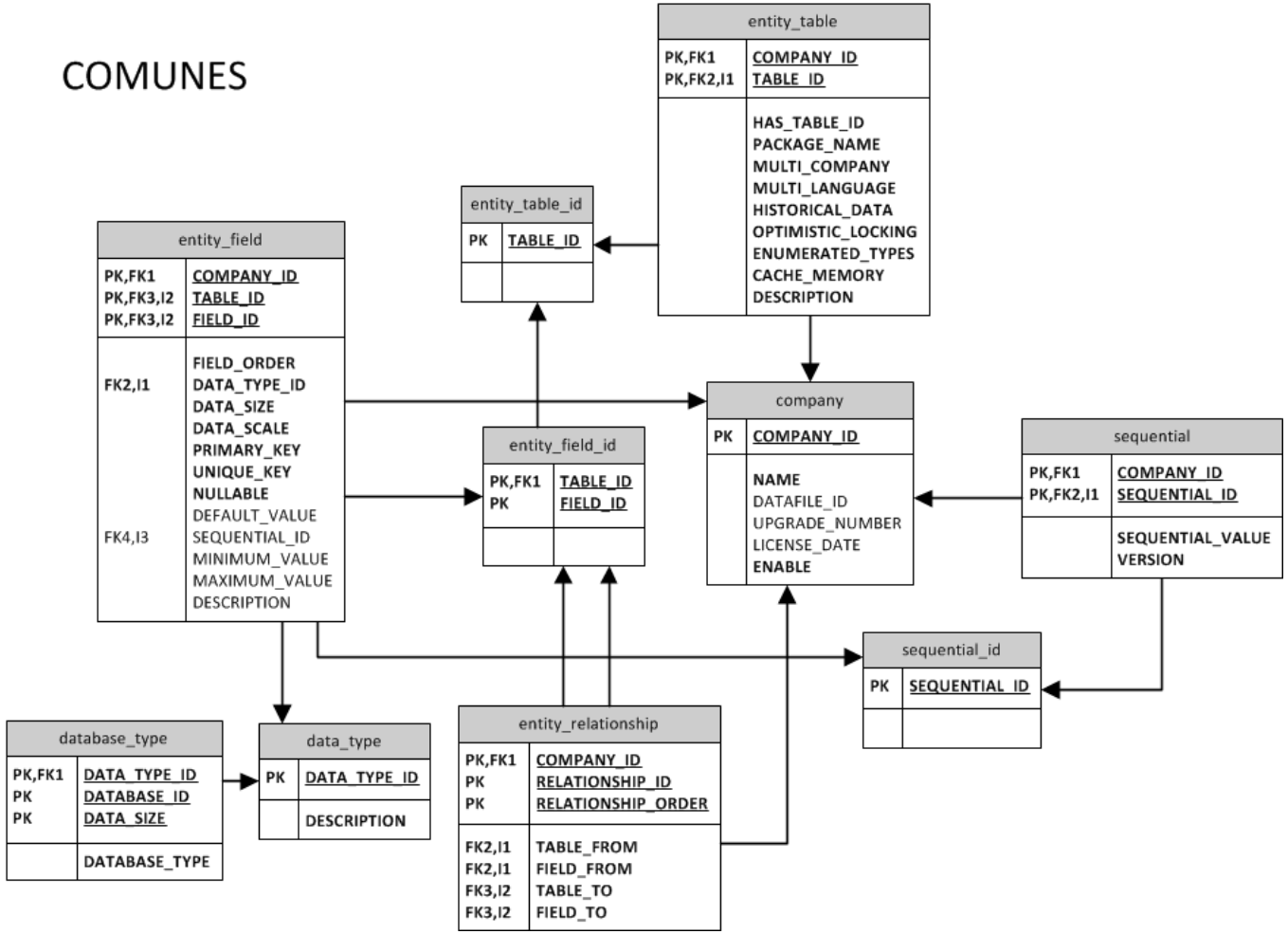


Proyecto: web.webxt



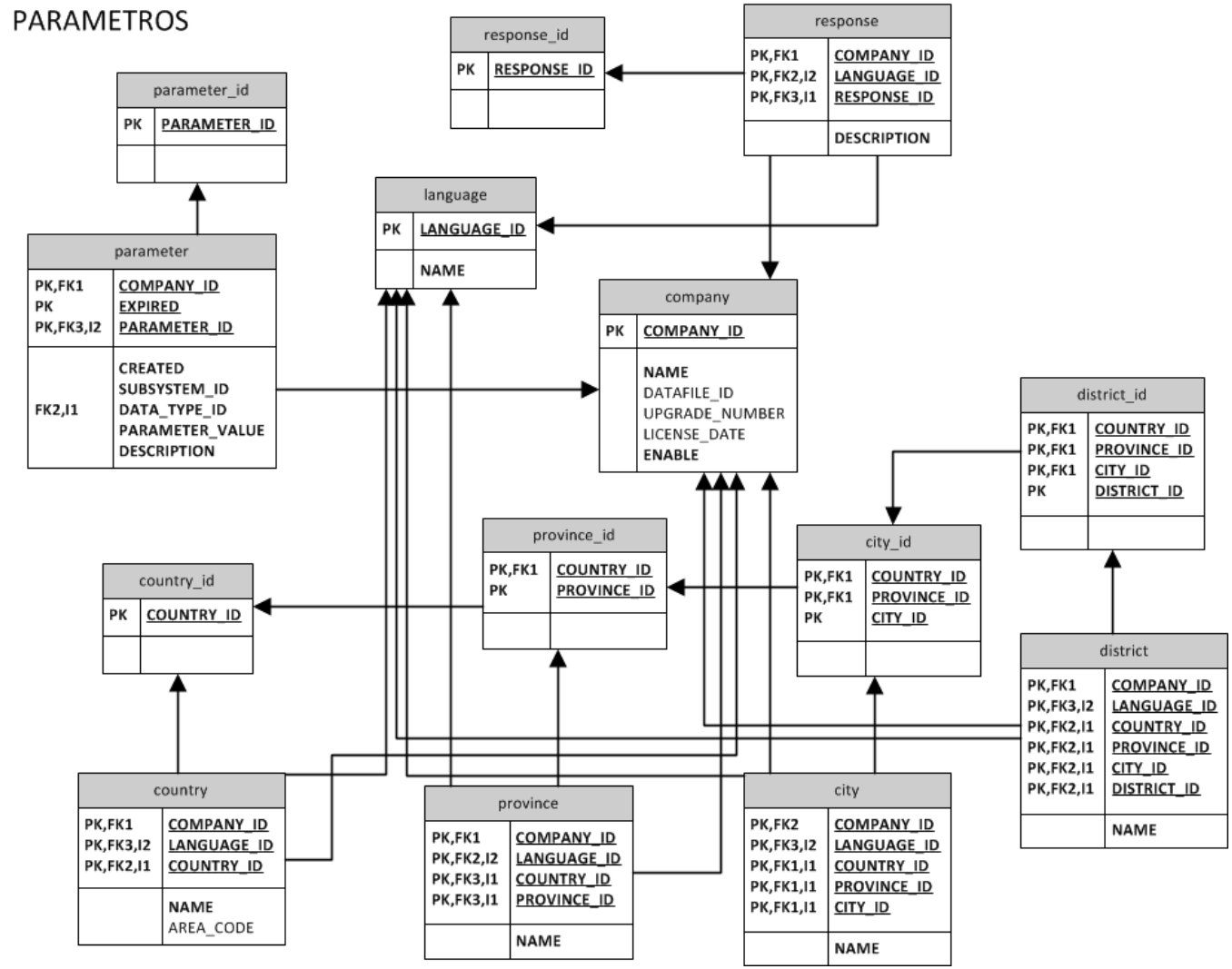
ANEXO 5. Diagramas Entidad-Relación.

Módulo: Comunes

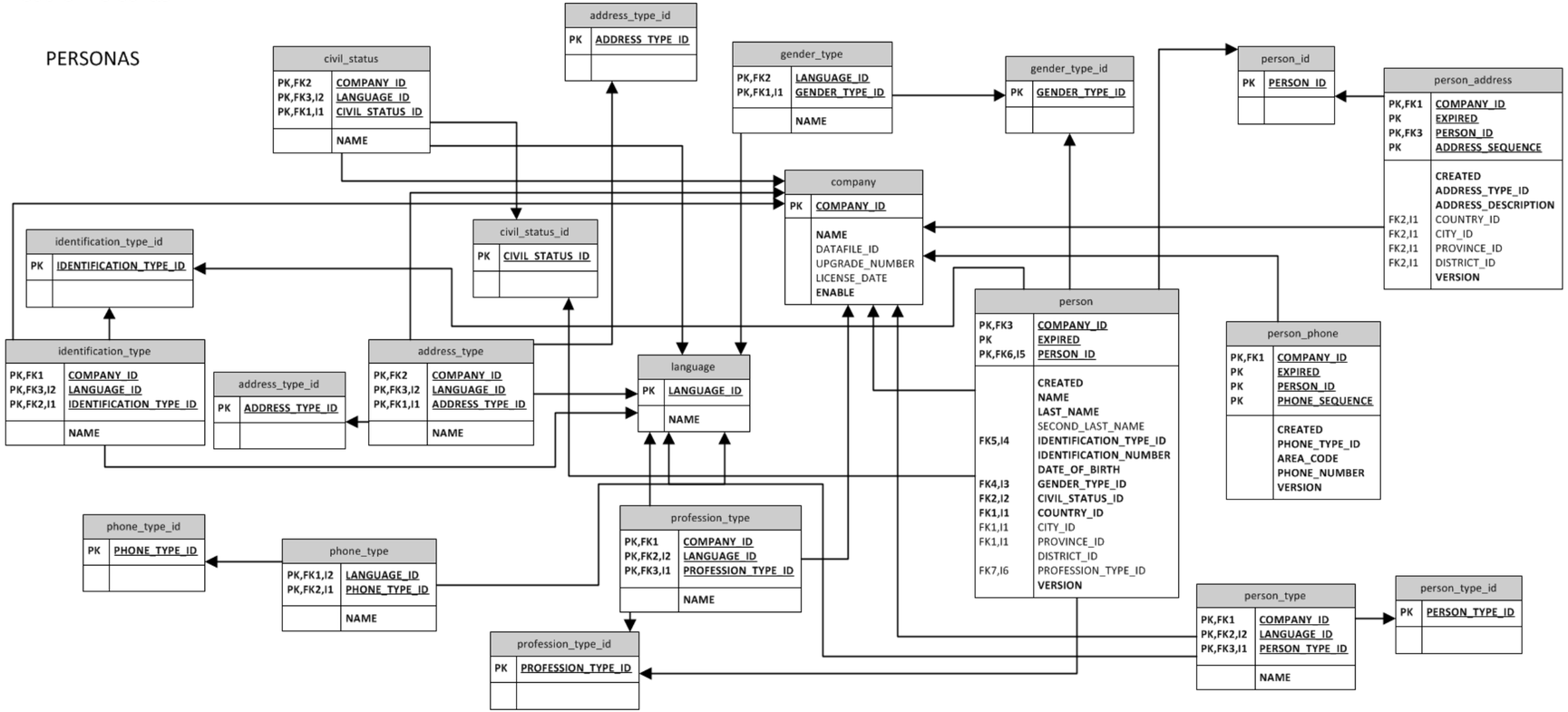


Módulo: Parámetros

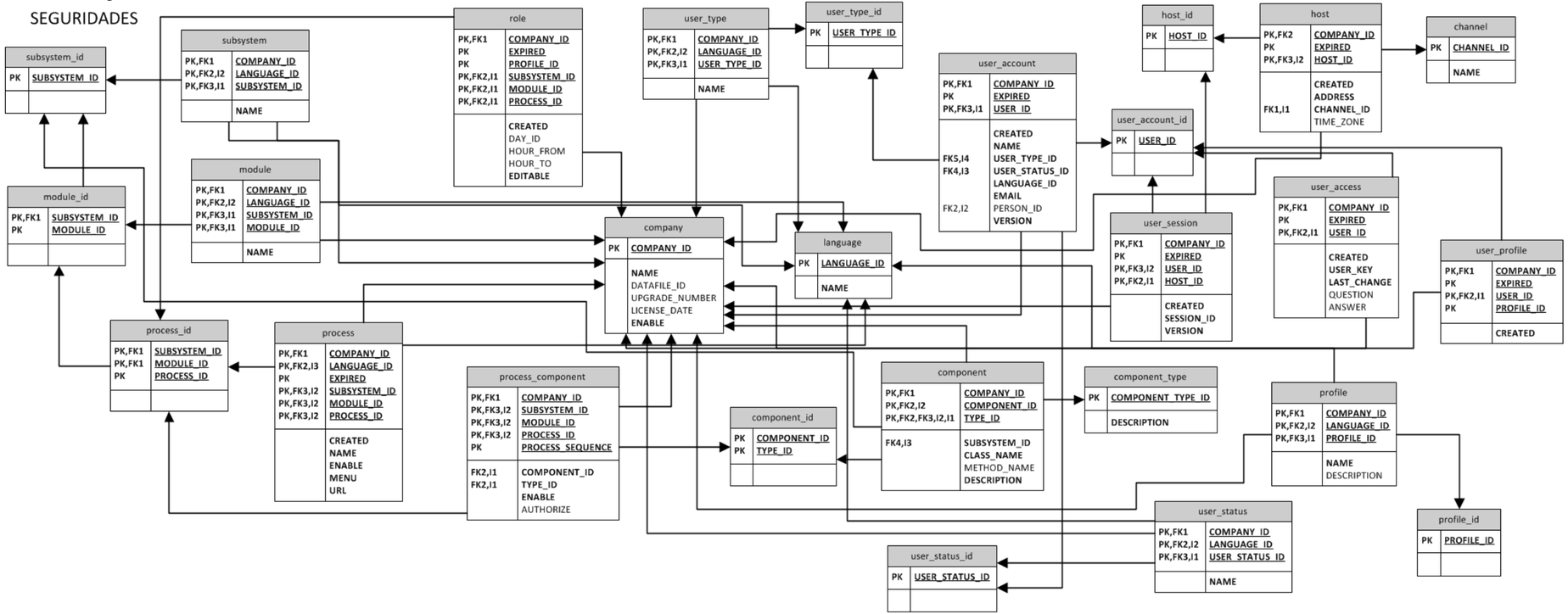
PARAMETROS


response_id
PK RESPONSE_ID
response
PK,FK1 COMPANY_ID
PK,FK2,I2 LANGUAGE_ID
PK,FK3,I1 RESPONSE_ID
DESCRIPTION
parameter_id
PK PARAMETER_ID
language
PK LANGUAGE_ID
NAME
parameter
PK,FK1 COMPANY_ID
PK EXPIRED
PK,FK3,I2 PARAMETER_ID
CREATED
SUBSYSTEM_ID
FK2,I1 DATA_TYPE_ID
PARAMETER_VALUE
DESCRIPTION
company
PK COMPANY_ID
NAME
DATAFILE_ID
UPGRADE_NUMBER
LICENSE_DATE
ENABLE
district_id
PK,FK1 COUNTRY_ID
PK,FK1 PROVINCE_ID
PK,FK1 CITY_ID
PK DISTRICT_ID
province_id
PK,FK1 COUNTRY_ID
PK PROVINCE_ID
city_id
PK,FK1 COUNTRY_ID
PK,FK1 PROVINCE_ID
PK CITY_ID
country_id
PK COUNTRY_ID
district
PK,FK1 COMPANY_ID
PK,FK3,I2 LANGUAGE_ID
PK,FK2,I1 COUNTRY_ID
PK,FK2,I1 PROVINCE_ID
PK,FK2,I1 CITY_ID
PK,FK2,I1 DISTRICT_ID
NAME
country
PK,FK1 COMPANY_ID
PK,FK3,I2 LANGUAGE_ID
PK,FK2,I1 COUNTRY_ID
NAME
AREA_CODE
province
PK,FK1 COMPANY_ID
PK,FK2,I2 LANGUAGE_ID
PK,FK3,I1 COUNTRY_ID
PK,FK3,I1 PROVINCE_ID
NAME
city
PK,FK2 COMPANY_ID
PK,FK3,I2 LANGUAGE_ID
PK,FK1,I1 COUNTRY_ID
PK,FK1,I1 PROVINCE_ID
PK,FK1,I1 CITY_ID
NAME

Módulo: Personas

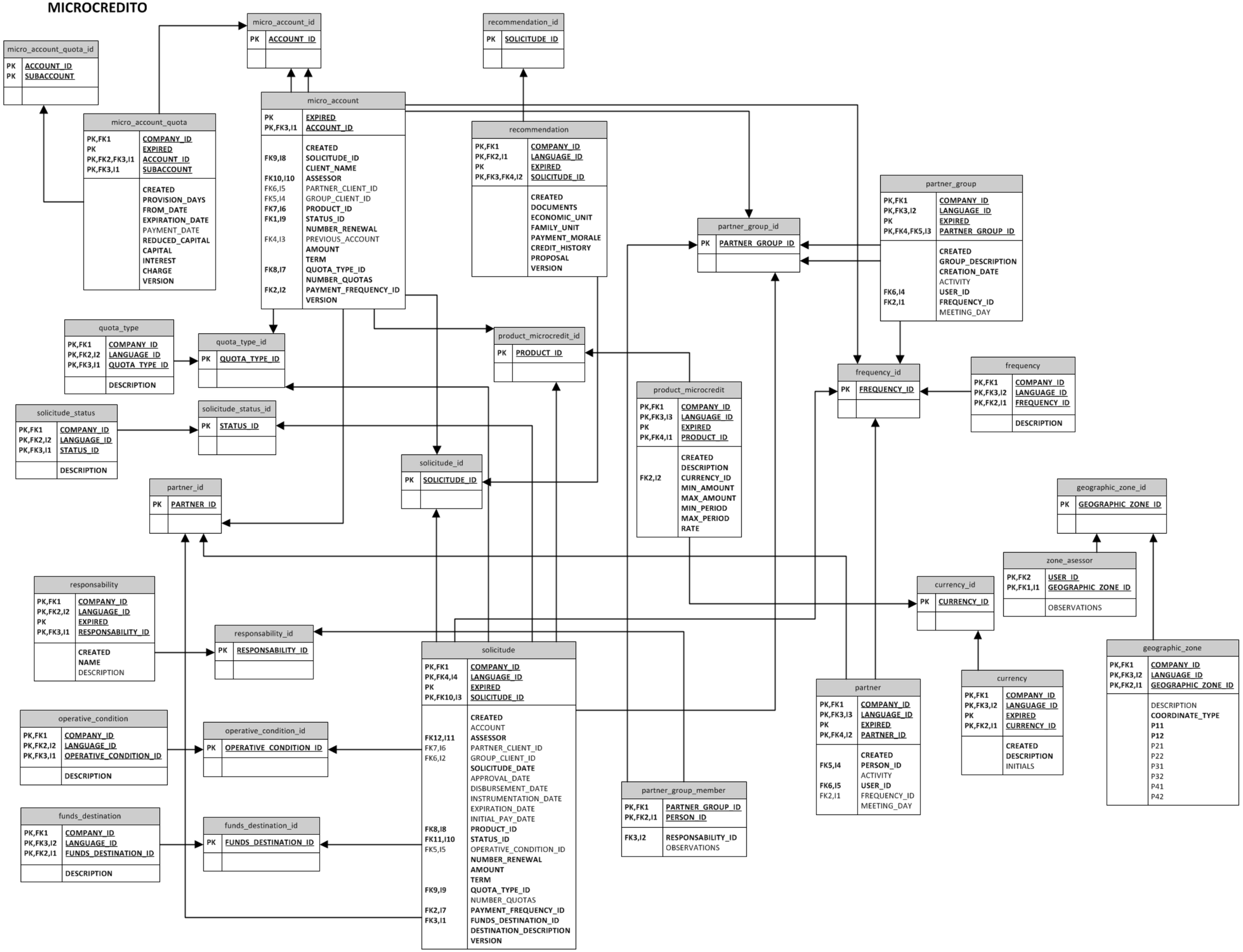


Módulo: Seguridades

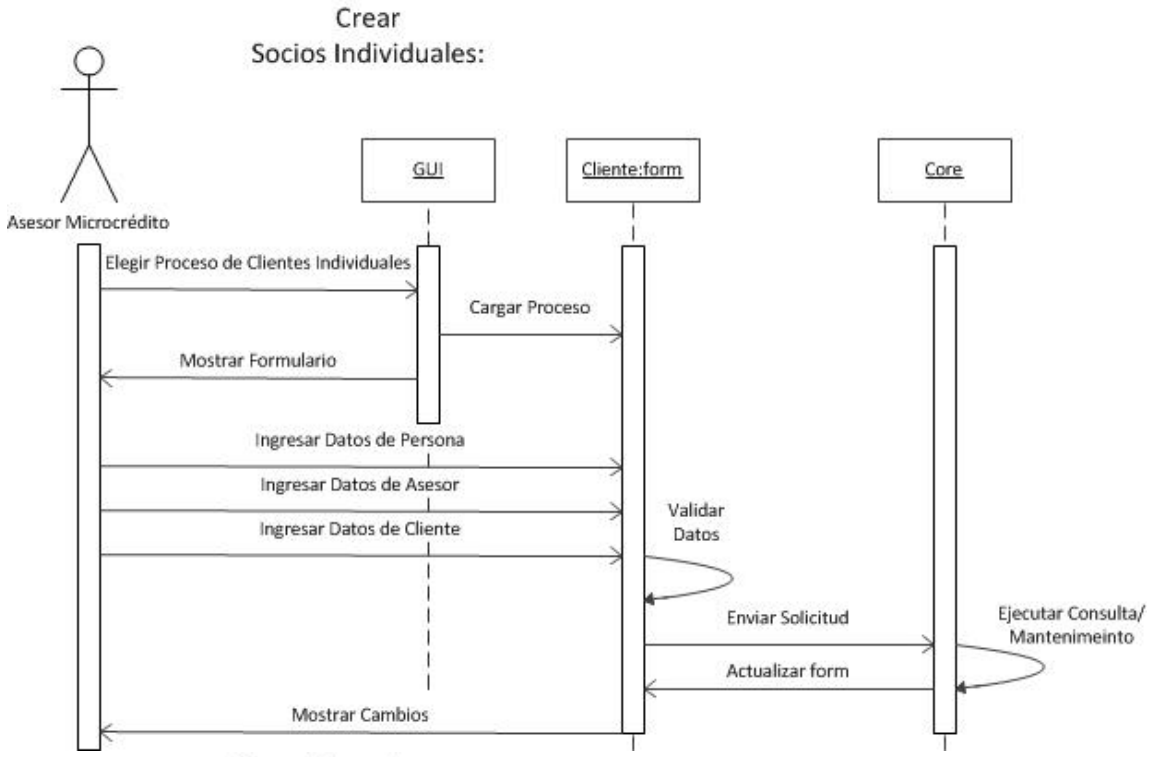


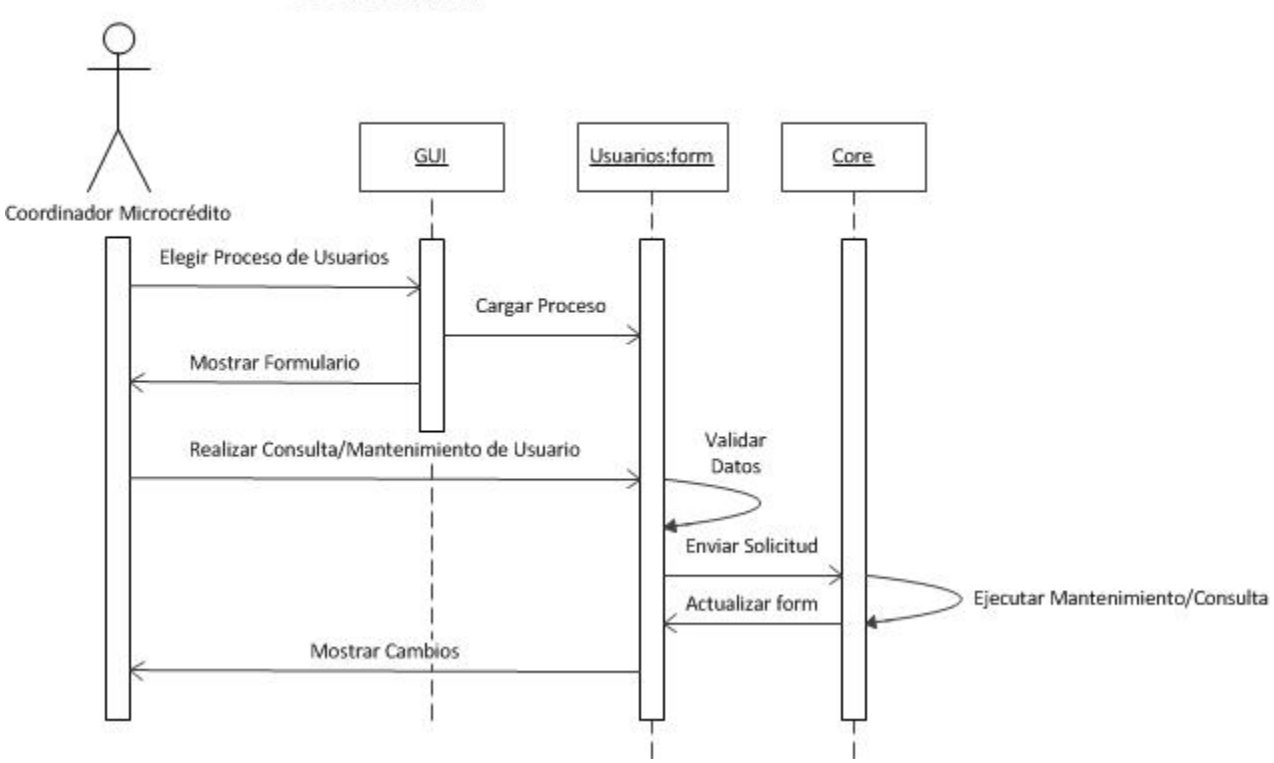
Módulo: Microcrédito

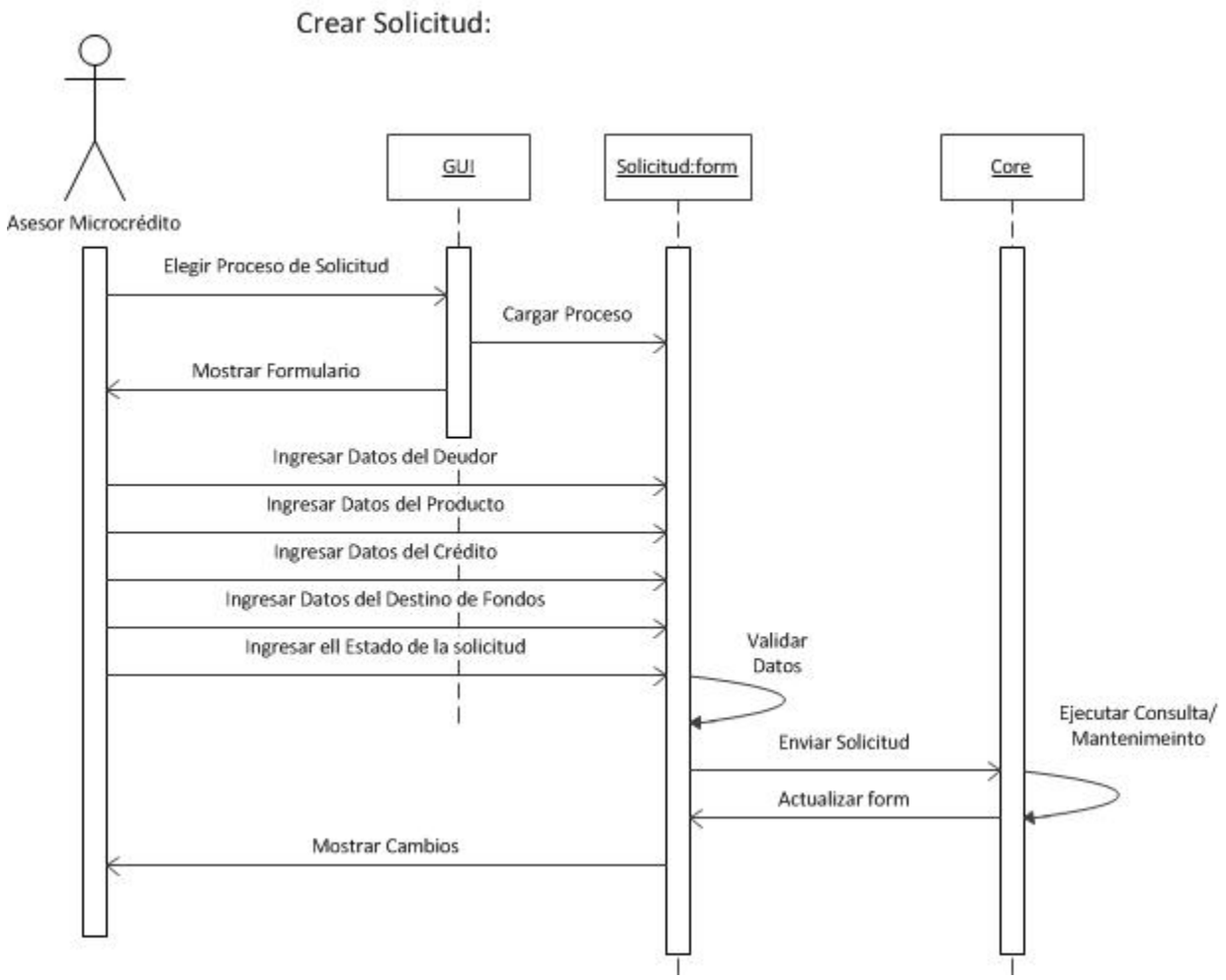
MICROCREDITO



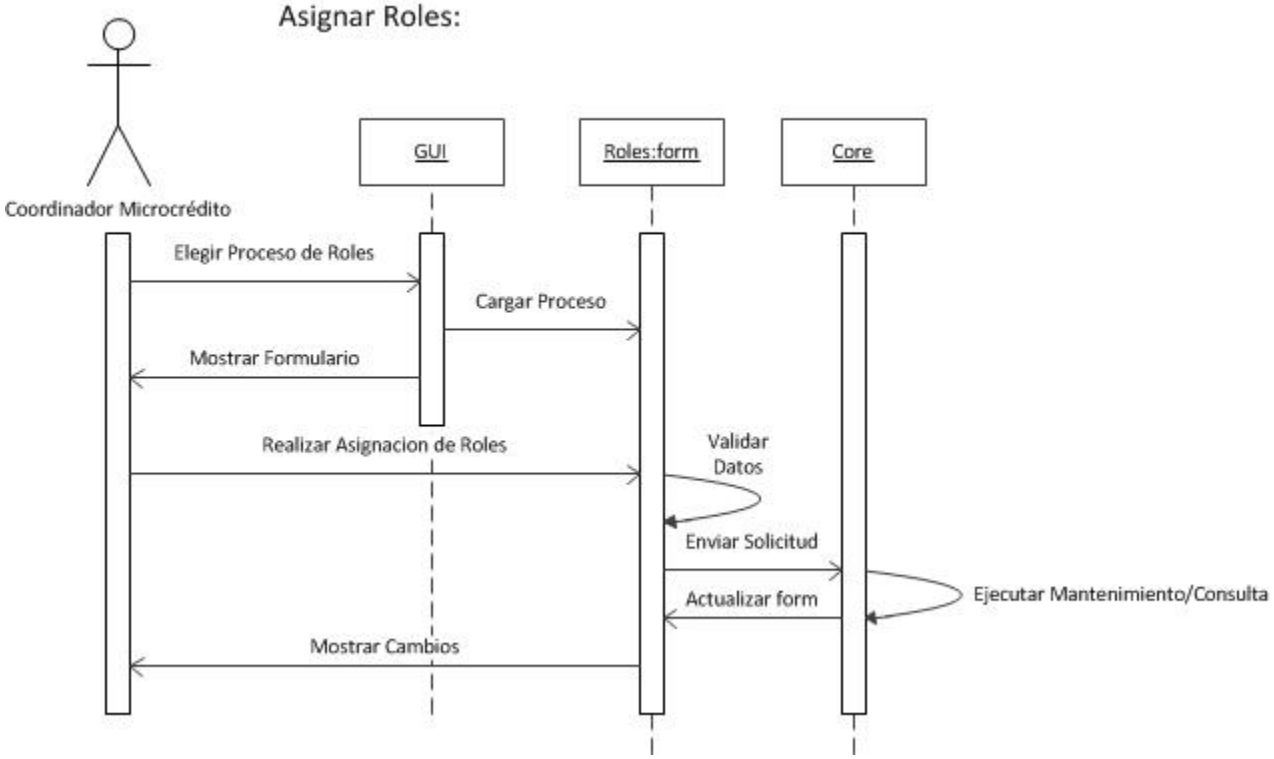
ANEXO 6. Diagramas de Secuencia

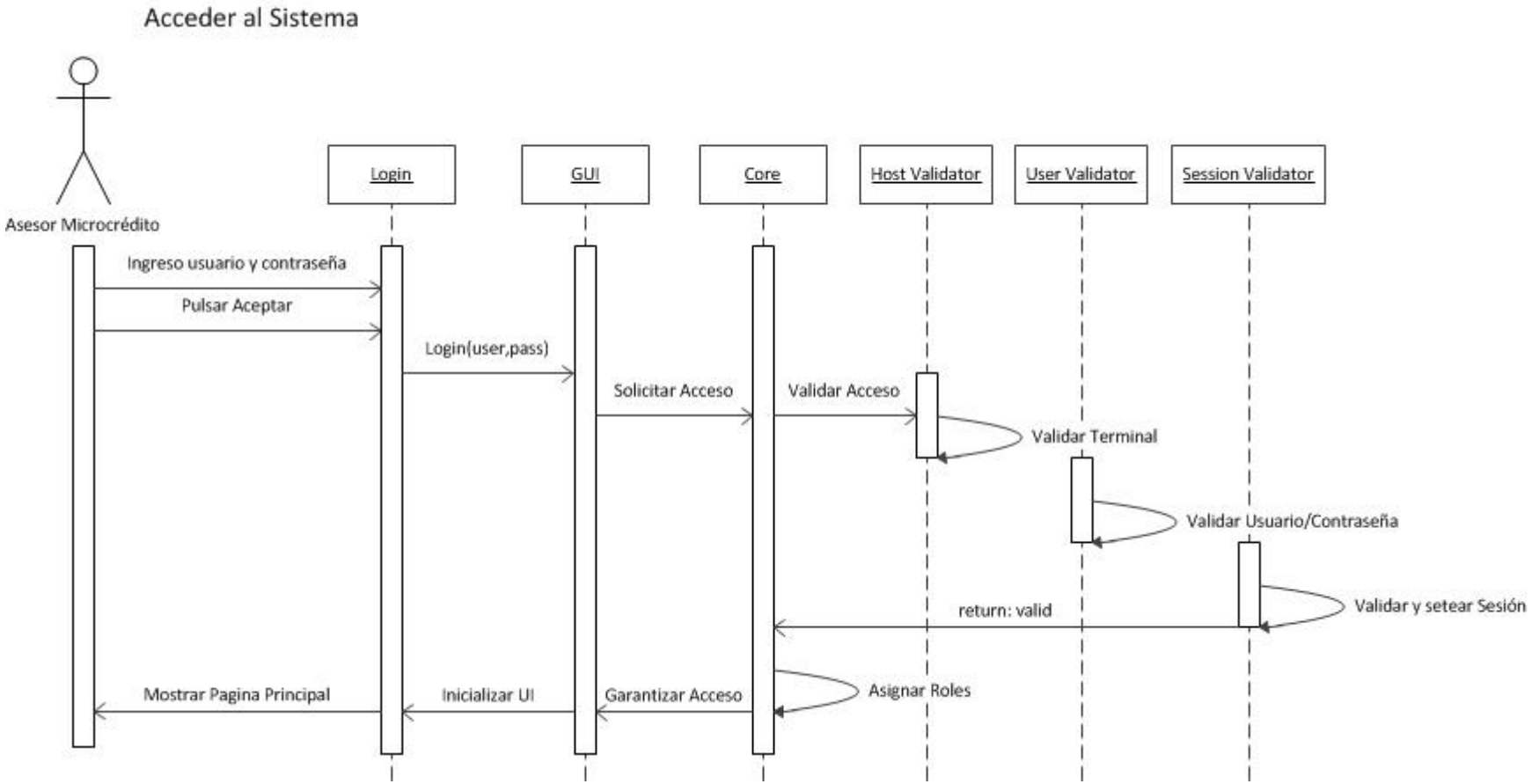


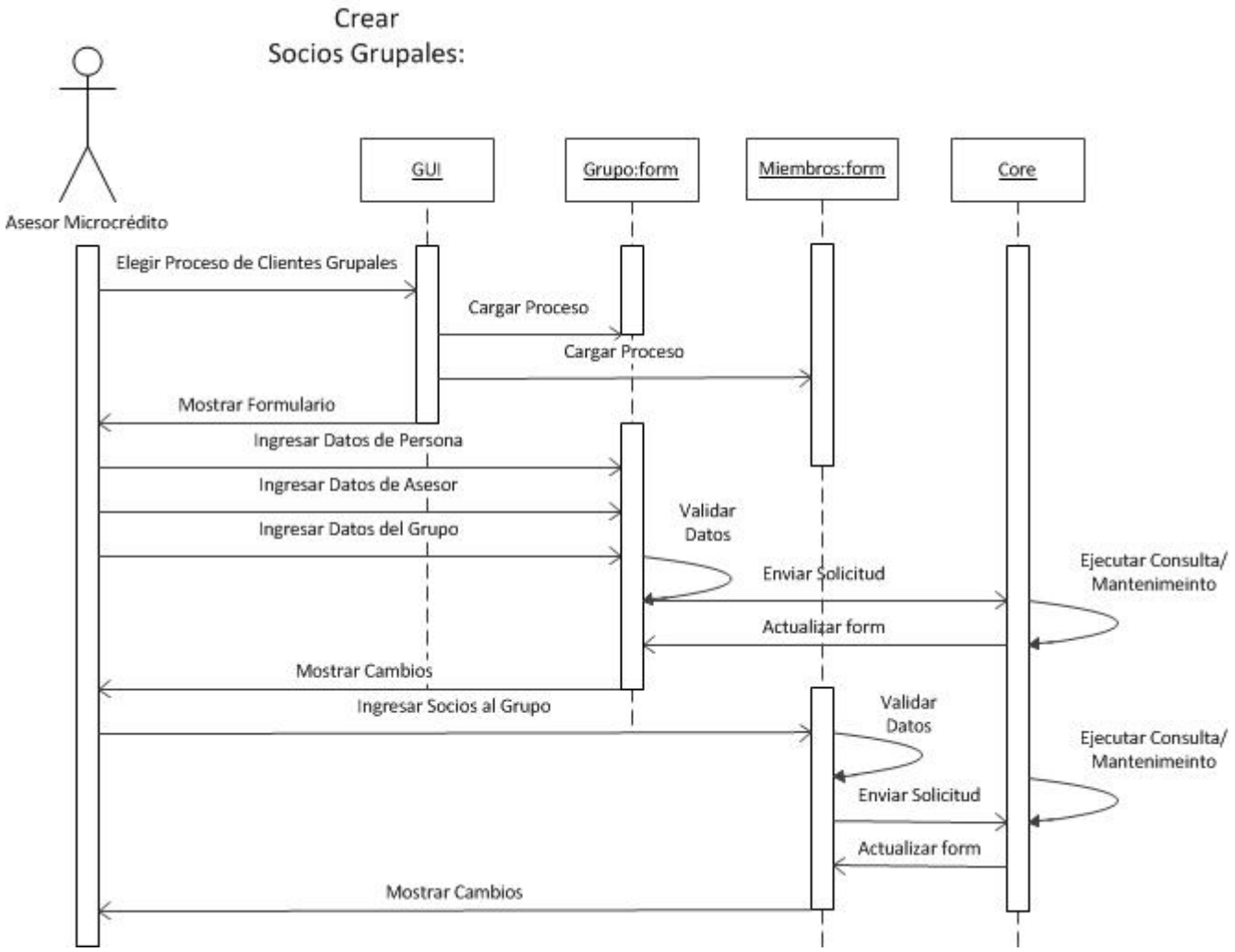



Crear Solicitud:
Asesor Microcrédito
GUI
Solicitud:form
Core
Elegir Proceso de Solicitud
Cargar Proceso
Mostrar Formulario
Ingresar Datos del Deudor
Ingresar Datos del Producto
Ingresar Datos del Crédito
Ingresar Datos del Destino de Fondos
Ingresar ell Estado de la solicitud
Validar Datos
Ejecutar Consulta/ Mantenimeinto
Enviar Solicitud
Actualizar form
Mostrar Cambios

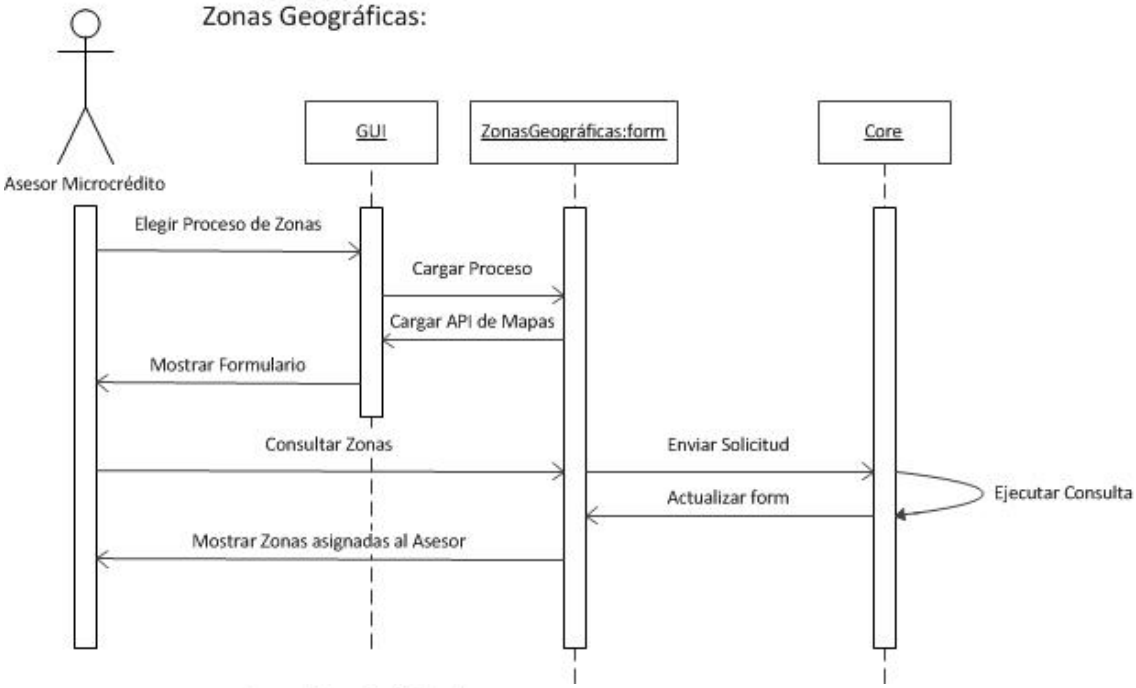
Asignar Roles:



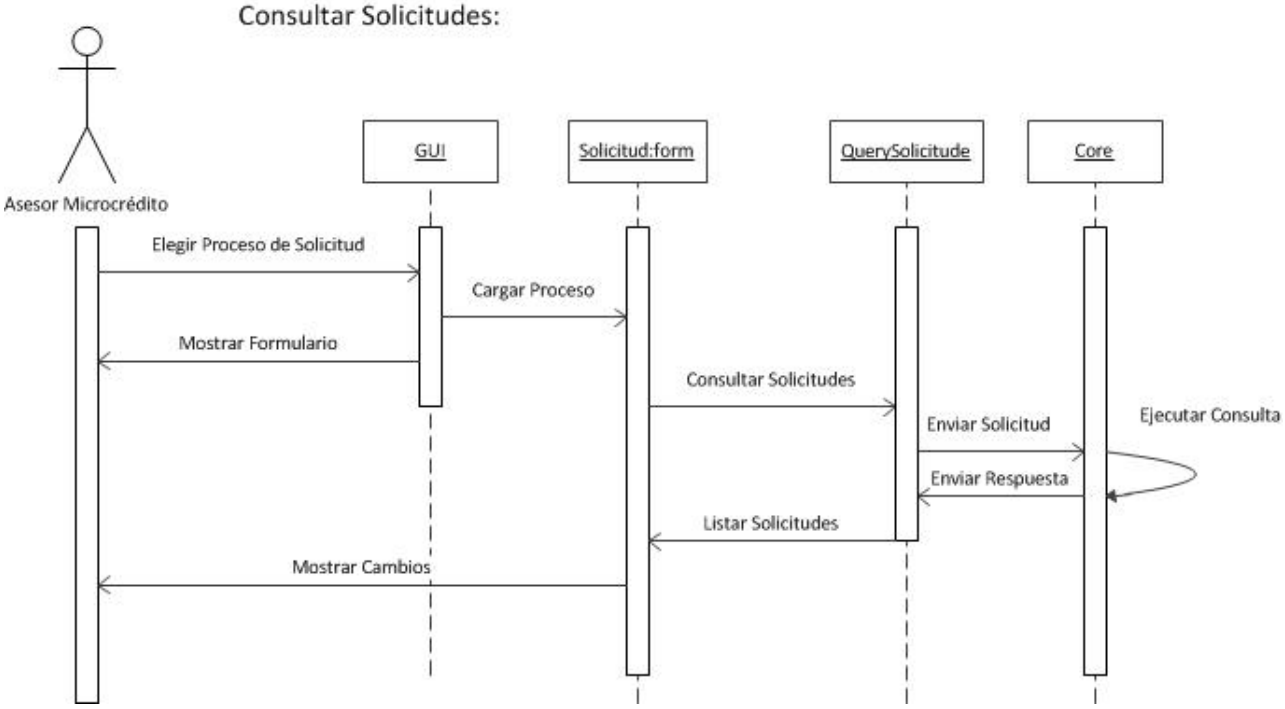

Acceder al Sistema
Asesor Microcrédito
Login
GUI
Core
Host Validator
User Validator
Session Validator
Ingreso usuario y contraseña
Pulsar Aceptar
Login(user,pass)
Solicitar Acceso
Validar Acceso
Validar Terminal
Validar Usuario/Contraseña
Validar y setear Sesión
return: valid
Asignar Roles
Garantizar Acceso
Inicializar UI
Mostrar Pagina Principal

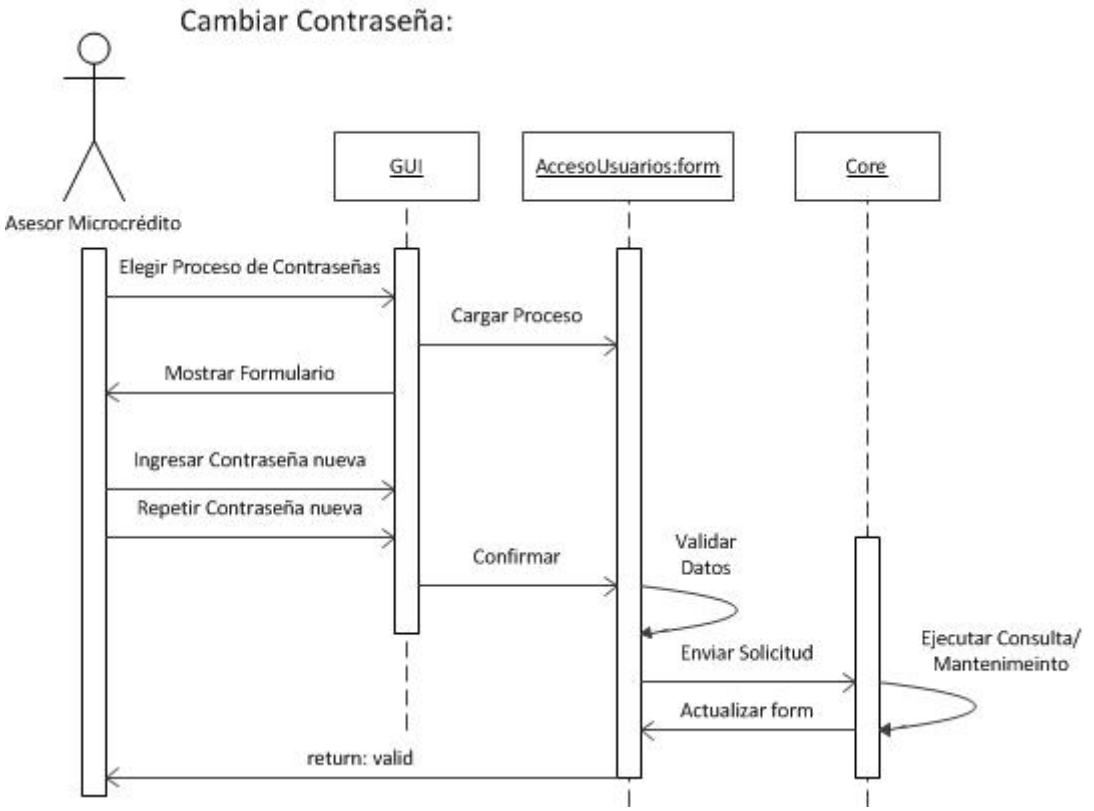

Crear
Socios Grupales:
Asesor Microcrédito
GUI
Grupo:form
Miembros:form
Core
Elegir Proceso de Clientes Grupales
Cargar Proceso
Cargar Proceso
Mostrar Formulario
Ingresar Datos de Persona
Ingresar Datos de Asesor
Ingresar Datos del Grupo
Validar Datos
Enviar Solicitud
Ejecutar Consulta/ Mantenimeinto
Actualizar form
Mostrar Cambios
Ingresar Socios al Grupo
Validar Datos
Ejecutar Consulta/ Mantenimeinto
Enviar Solicitud
Actualizar form
Mostrar Cambios

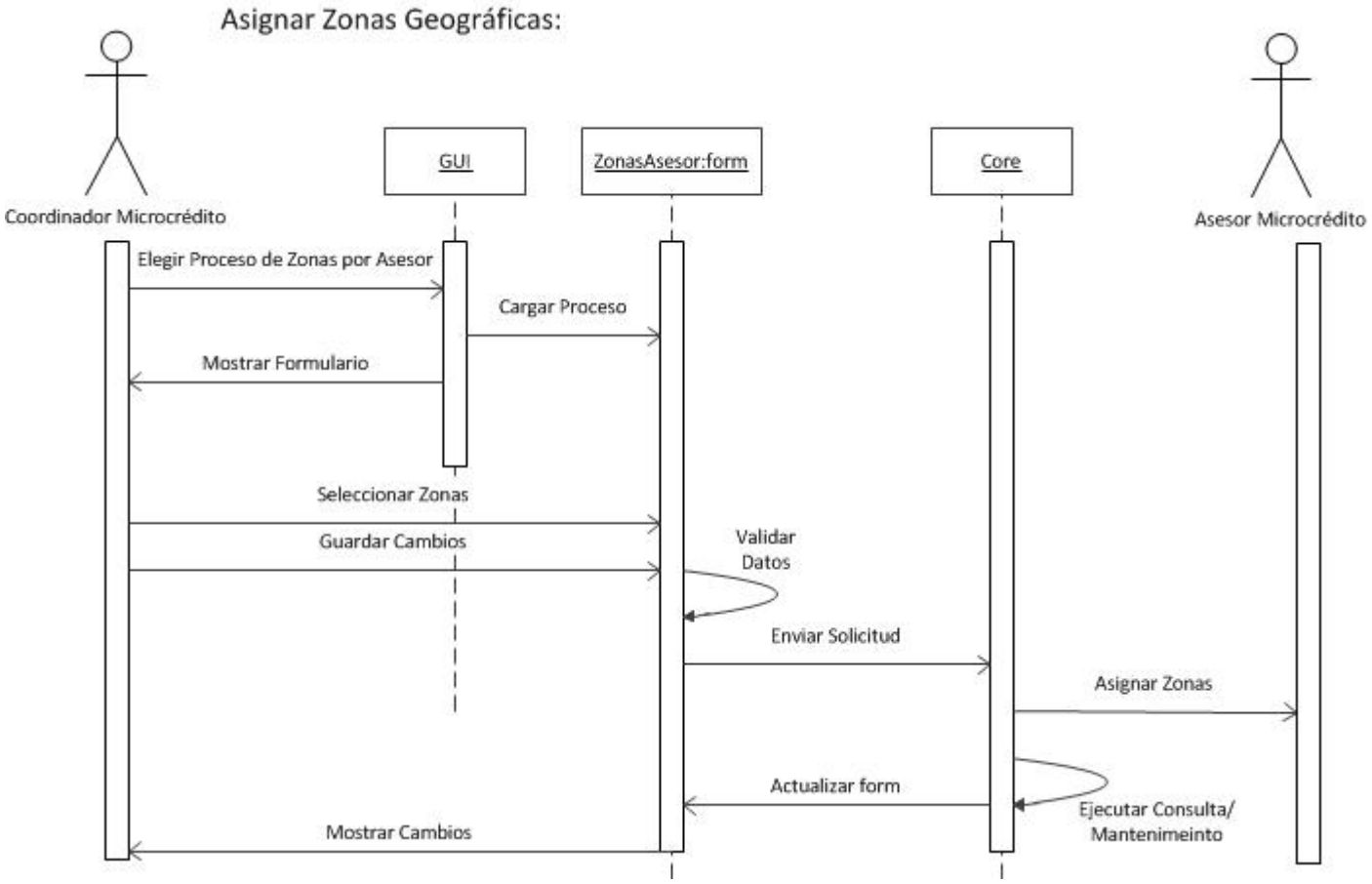
Consultar
Zonas Geográficas:

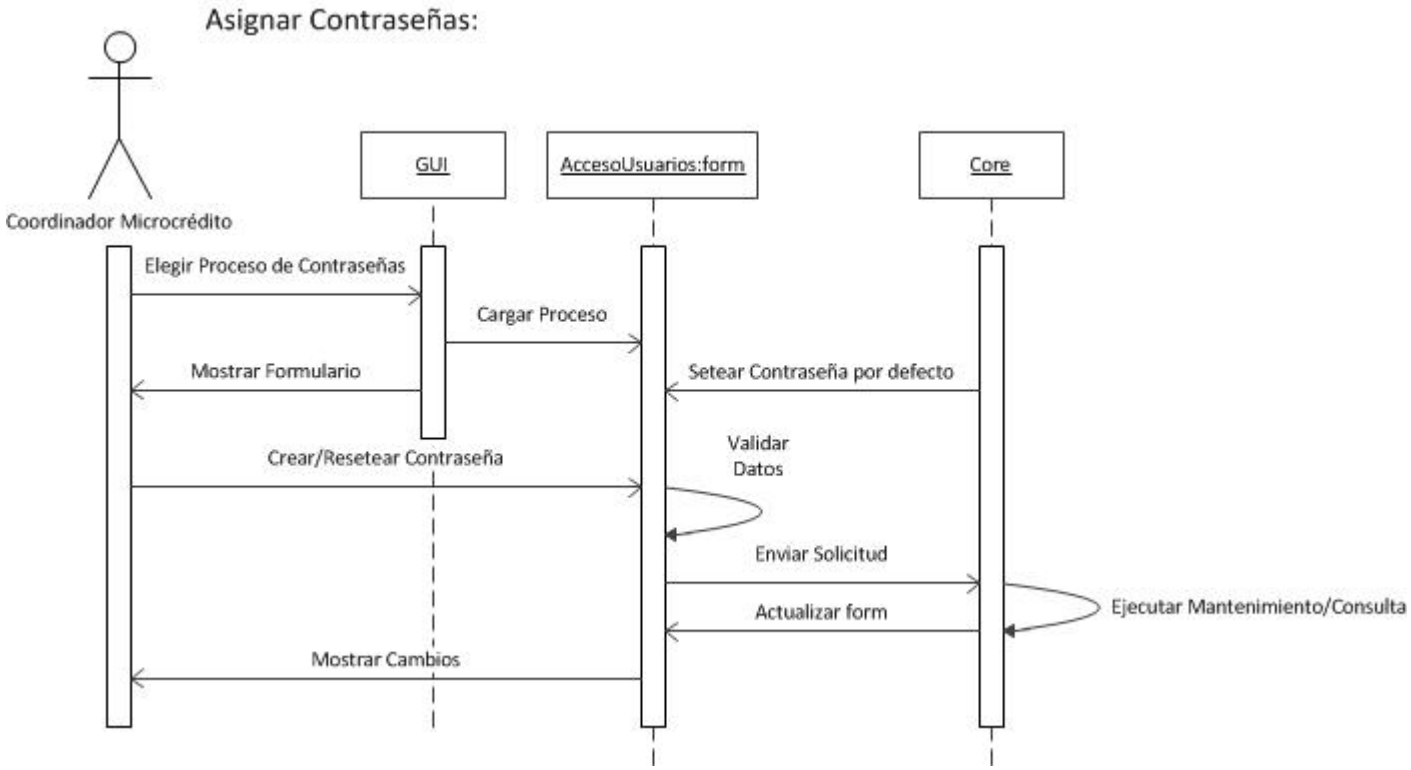


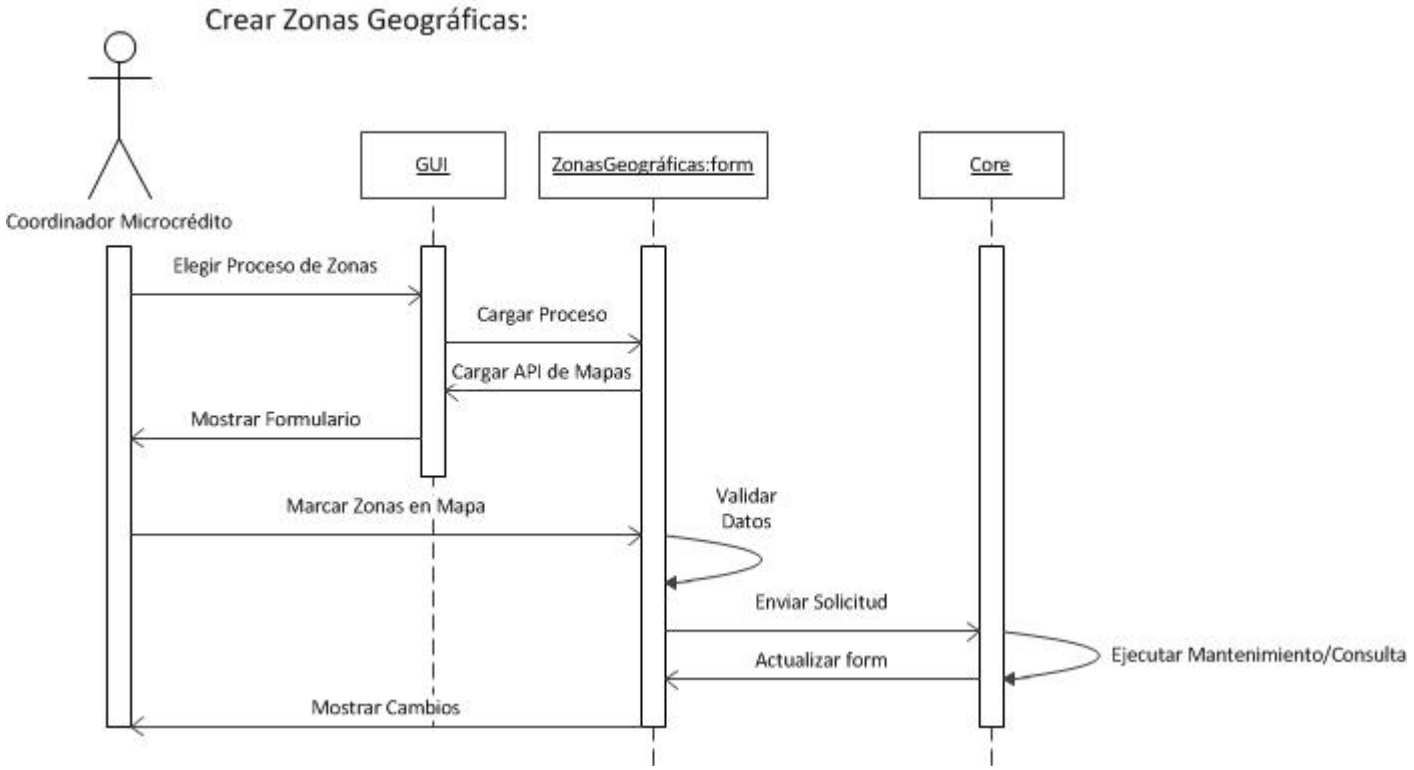
Consultar Solicitudes:




Cambiar Contraseña:
Asesor Microcrédito
GUI
AccesoUsuarios:form
Core
Elegir Proceso de Contraseñas
Cargar Proceso
Mostrar Formulario
Ingresar Contraseña nueva
Repetir Contraseña nueva
Confirmar
Validar Datos
Enviar Solicitud
Ejecutar Consulta/ Mantenimeinto
Actualizar form
return: valid


Asignar Zonas Geográficas:
Coordinador Microcrédito
GUI
ZonasAsesor:form
Core
Asesor Microcrédito
Elegir Proceso de Zonas por Asesor
Cargar Proceso
Mostrar Formulario
Seleccionar Zonas
Guardar Cambios
Validar Datos
Enviar Solicitud
Asignar Zonas
Actualizar form
Ejecutar Consulta/ Mantenimeinto
Mostrar Cambios


Asignar Contraseñas:
Coordinador Microcrédito
GUI
AccesoUsuarios:form
Core
Elegir Proceso de Contraseñas
Cargar Proceso
Mostrar Formulario
Setear Contraseña por defecto
Crear/Resetear Contraseña
Validar
Datos
Enviar Solicitud
Actualizar form
Ejecutar Mantenimiento/Consulta
Mostrar Cambios


Crear Zonas Geográficas:
Coordinador Microcrédito
GUI
ZonasGeográficas:form
Core
Elegir Proceso de Zonas
Cargar Proceso
Cargar API de Mapas
Mostrar Formulario
Marcar Zonas en Mapa
Validar Datos
Enviar Solicitud
Actualizar form
Ejecutar Mantenimiento/Consulta
Mostrar Cambios

ANEXO 7. POM de configuración del Proyecto *webxt*

```
<project      xmlns="http://maven.apache.org/POM/4.0.0"      xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
      xsi:schemaLocation="http://maven.apache.org/maven-v4_0_0.xsd">
      <modelVersion>4.0.0</modelVersion>
      <groupId>mobile</groupId>
      <artifactId>mobile</artifactId>
      <packaging>pom</packaging>
      <version>2.0</version>
      <name>MOBILE</name>
         <properties>
           <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
      </properties>

      <modules>
            <module>tools</module>
            <module>common</module>
            <module>core</module>
            <module>logic</module>
            <module>entity</module>
            <module>web</module>
      </modules>
      <build>
            <plugins>
                  <plugin>
                        <groupId>org.apache.maven.plugins</groupId>
                        <artifactId>maven-compiler-plugin</artifactId>
                        <version>2.3</version>
                        <configuration>
                              <source>1.6</source>
                              <target>1.6</target>
```

```
                        <encoding>UTF-8</encoding>
                        <showDeprecation>true</showDeprecation>
                              <showWarnings>true</showWarnings>
                  </configuration>
            </plugin>
            <plugin>
                  <groupId>org.apache.maven.plugins</groupId>
                  <artifactId>maven-resources-plugin</artifactId>
                  <version>2.5</version>
                  <configuration>
                    <encoding>UTF-8</encoding>
                  </configuration>
            </plugin>
            <plugin>
                  <groupId>org.apache.maven.plugins</groupId>
                  <artifactId>maven-javadoc-plugin</artifactId>
                  <version>2.7</version>
                  <configuration>
                        <doclet>org.umlgraph.doclet.UmlGraphDoc</doclet>
                        <docletArtifact>
                              <groupId>org.umlgraph</groupId>
                              <artifactId>doclet</artifactId>
                              <version>5.1</version>
                        </docletArtifact>
                        <additionalparam>-views</additionalparam>
                        <show>private</show>
                        <useStandardDocletOptions>true</useStandardDocletOptions>
                  </configuration>
            </plugin>
      </plugins>
  </build>
</project>
```

ANEXO 8. Archivo persistence.xml de la configuración para JPA

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0"
      xmlns="http://java.sun.com/xml/ns/persistence"        xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
      <persistence-unit name="central" transaction-type="RESOURCE_LOCAL">
            <jar-file>mobile.entity.common-2.0.jar</jar-file>
            <jar-file>mobile.entity.parameter-2.0.jar</jar-file>
            <jar-file>mobile.entity.person-2.0.jar</jar-file>
            <jar-file>mobile.entity.security-2.0.jar</jar-file>
            <jar-file>mobile.entity.microcredit-2.0.jar</jar-file>
            <exclude-unlisted-classes>false</exclude-unlisted-classes>
            <properties>
<!--Local connection -->
<property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/microxt" />
<property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
<property name="javax.persistence.jdbc.user" value="root" />
<property name="javax.persistence.jdbc.password" value="" />

<!--MySQL -->
      <property name="eclipselink.target-database"
                        value="org.eclipse.persistence.platform.database.MySQLPlatform" />
      <!--Logging -->
      <property name="eclipselink.logging.level" value="FINE" />
      <property name="eclipselink.logging.timestamp" value="false" />
      <property name="eclipselink.logging.session" value="false" />
      <property name="eclipselink.logging.thread" value="false" />
```

```
      <property name="eclipselink.logging.exceptions" value="true" />
      <property name="eclipselink.logging.connection" value="true" />
      </properties>
      </persistence-unit>
      <persistence-unit name="generator" transaction-type="RESOURCE_LOCAL">
            <jar-file>mobile.entity.common-2.0.jar</jar-file>
            <exclude-unlisted-classes>false</exclude-unlisted-classes>
            <properties>
                  <!--Local connection -->
                  <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/microxt"
/>
            <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
      <property name="javax.persistence.jdbc.user" value="root" />
      <property name="javax.persistence.jdbc.password" value="" />
      <!--MySQL -->
      <property name="eclipselink.target-database"
      value="org.eclipse.persistence.platform.database.MySQLPlatform"/>
      <!--Logging -->
      <property name="eclipselink.logging.level" value="FINE" />
      <property name="eclipselink.logging.timestamp" value="false" />
      <property name="eclipselink.logging.session" value="false" />
      <property name="eclipselink.logging.thread" value="false" />
      <property name="eclipselink.logging.exceptions" value="true" />
      <property name="eclipselink.logging.connection" value="true" />
      </properties>
      </persistence-unit>
</persistence>
```

ANEXO 9. Archivo web.xml para el CORE server.

```
<?xml version = '1.0' encoding = 'windows-1252'?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee       http://java.sun.com/xml/ns/javaee/web-
app_2_5.xsd">
      <description></description>
      <!-- Application Startup Class -->
      <listener>
            <listener-class>mobile.core.server.PersistenceListener</listener-class>
      </listener>
      <servlet>
            <servlet-name>Core</servlet-name>
            <servlet-class>mobile.core.server.Core</servlet-class>
            <!--<load-on-startup>1</load-on-startup>-->
      </servlet>
      <servlet-mapping>
            <servlet-name>Core</servlet-name>
            <url-pattern>/Core</url-pattern>
      </servlet-mapping>
      <session-config>
            <session-timeout>35</session-timeout>
      </session-config>
</web-app>
```

ANEXO 10. Archivo web.xml para el CORE simulator.

```
<?xml version = '1.0' encoding = 'windows-1252'?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee       http://java.sun.com/xml/ns/javaee/web-
app_2_5.xsd">
      <description></description>
      <!-- Application Startup Class -->
      <!-- <listener>
            <listener-class>mobile.core.simulator.PersistenceListener</listener-class>
      </listener> -->
      <servlet>
            <servlet-name>Simulador</servlet-name>
            <servlet-class>mobile.core.simulator.Simulator</servlet-class>
            <!--<load-on-startup>1</load-on-startup>-->
      </servlet>
      <servlet-mapping>
            <servlet-name>Simulador</servlet-name>
            <url-pattern>/Simulador</url-pattern>
      </servlet-mapping>
      <session-config>
            <session-timeout>35</session-timeout>
      </session-config>
</web-app>
```

ANEXO 11. Archivo Mobile.gwt.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<module rename-to='webxt'>
      <!-- Inherit the core Web Toolkit stuff. -->
      <inherits name='com.google.gwt.user.User' />
      <inherits name='com.google.gwt.http.HTTP' />
      <inherits name='com.google.gwt.maps.GoogleMaps' />
      <inherits name='com.googlecode.gwt.crypto.Crypto'/>
      <inherits name="com.google.gwt.i18n.I18N"/>

      <!-- Inherit the default GWT style sheet -->
      <inherits name='com.google.gwt.user.theme.standard.Standard' />

      <!-- GXT -->
      <inherits name='com.extjs.gxt.ui.GXT' />

      <!-- Locale -->
      <extend-property name="locale" values="es"/>
      <set-property-fallback name="locale" value="es"/>

      <!-- Mobile inherits -->
      <inherits name="mobile.common.Message" />
      <inherits name="mobile.common.Tools" />

      <!-- Specify the app entry point class. -->
      <entry-point class='mobile.web.webxt.client.Mobile' />

      <!-- Specify the paths for translatable code -->
      <source path='client' />
      <source path='shared' />
</module>
```